PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04B

A2

(11) International Publication Number: **WO 99/09658**

(43) International Publication Date: 25 February 1999 (25.02.99)

(21) International Application Number: PCT/US98/16894

(22) International Filing Date: 14 August 1998 (14.08.98)

(30) Priority Data:

60/055,782	15 August 1997 (15.08.97)	US
60/057,256	29 August 1997 (29.08.97)	US
60/060,612	1 October 1997 (01.10.97)	US
08/970,894	13 November 1997 (13.11.97)	US
60/065,416	13 November 1997 (13.11.97)	US
08/971,002	14 November 1997 (14.11.97)	US

(71) Applicant: INERGY ONLINE, INC. [US/US]; 20 Mall Road, Burlington, MA 01803 (US).

(72) Inventor: BELANGER, Charles, E.; 345 Boston Road, Billerica, MA 01821 (US).

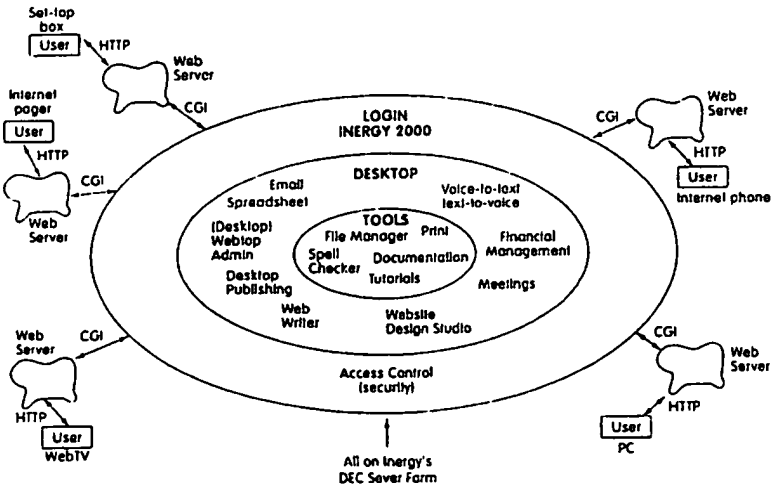
(74) Agents: CELLA, Charles, H. et al.; Foley, Hoag & Eliot LLP, One Post Office Square, Boston, MA 02109 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: SERVER-SIDED INTERNET-BASED PLATFORM INDEPENDENT OPERATING SYSTEM AND APPLICATION SUITE



(57) Abstract

The systems and methods described herein provide different types of Web authoring, Web site management, and communication software technology, including but not limited to full multimedia authoring, online libraries, sounds, forms, e-mail, facsimile, voice-mail, pager, telephone, financial management, true document printing (as opposed to screen printing), text-to-voice and voice-to-text conversion, file management, spreadsheets, all accessed and run via the Internet. The system resides entirely on an Internet Web Server site and interacts with users via conventional progamming languages written for a universal protocol. As a result, there is no need for client-side messaging software. All software is provided on the server side. The only software the user needs is any form of communications module and an electronic communications connection. Because the system is platform and operating system independent, a user may author, create, maintain, send and receive messages from any platform, using any conventional operating system. A user may customize their desktop configuration and may run a variety of different applications. Moreover, a user may switch between applications, and transfer text, graphics, or sound files between applications.

SERVER-SIDED INTERNET-BASED PLATFORM INDEPENDENT OPERATING SYSTEM AND APPLICATION SUITE

Cross-Reference to Related Applications

This application is related to U.S. provisional patent application serial no. 60/030,994, entitled, "Remote Communication Management System", filed November 15, 1996; U.S. provisional patent application serial no. 60/030,996, entitled, "Remote Home Page Authoring System", filed November 15, 1996; U.S. provisional patent application serial no. 60/038,411, entitled, "Server-Sided Technology for Remote Television Computerization", filed February 18, 1997; U.S. provisional patent application, serial number 60/055,782, entitled, "Server-Sided Internet Based Operating System", filed August 15, 1997; U.S. provisional patent application, serial number 60/057,256, entitled "Server-Sided Web Based Operating System and Desktop Manager," filed August 29, 1997, U.S. provisional patent application, serial number 60/060,612, entitled "Server-Sided Internet-Based Platform Independent Operating System and Application Suite," filed October 1, 1997, U.S. provisional patent application, serial number 60/065,416, entitled "Server-Sided Internet-Based Platform Independent Operating System and Application Suite," filed November 13, 1997, U.S. utility patent application number 08/970,894, entitled "Remote Communication and Information Management System," filed November 14, 1997, and U.S. utility patent application number 08/971,002, entitled "Remote Home Page Authorizing System," filed November 14, 1997, all of which are pending and are incorporated herein by reference.

Background of The Invention

The evolution of the computer industry has been from mainframes, where all users have to take turns running software programs on a central computer system from "dumb" terminals on their desks, to smart and powerful desktop personal computers (PCs) in which users run software either from a removable disk or a hard drive.

As users have demanded numerous enhanced features for the software programs, it has become increasingly complex and difficult to install, maintain and run programs from local storage. For example, a typical word processing program fills several disks, even in a compressed format. This large size, combined with distribution challenges and costs, may make it impractical and not cost-effective to upgrade the programs on a frequent basis. Also, the programs have to be installed and configured by each user on their own PC, which can be challenging and time consuming. An additional disadvantage of PCs is that they are not very mobile, and powerful mobile laptops are very expensive.

Networks of PCs, typically in an office setting, have allowed key programs to be run from a central server without requiring users to take turns using the software. However, if a user wishes to be able to run several different programs simultaneously, and to switch between the programs at will, the user must have a powerful operating system installed on the PC, such as the MICROSOFT WINDOWS operating system, which takes up a significant amount of storage space on the hard drive. Also, in some conventional systems, each application must share a common operating system, limiting the types of applications available to the user of such operating systems.

One of the most significant developments in the computer industry has been the growth of the Internet and the World Wide Web. Computer users view Web sites, create and manage Web sites, and review and send electronic mail messages. All of these functions require software installed on the user's PC. The PC ties the user to a particular location. Moreover, such functions cannot be provided by many less-powerful computing devices or non-computing appliances. Less powerful devices do not have adequate local storage to permit the user to install powerful programs, such as web authoring programs.

None of the software on the application programs on the market today, such as WORDPERFECT, MICROSOFT WORD, etc. can run on network computer ("NC") appliances. Yet all of these NC appliances may be configured to allow a user to access the Internet. The problem is severe limitations on what they can do once connected to the Internet.

Summary Of The Invention

The present invention may be better understood by reference to a number of commonly used terms, definitions of which are as follows:

The term "client," as used herein, encompasses any data processing systems suitable for establishing a communication link to a communications network, such as an Internet site. An Internet site can be any program running on a data processing platform that connects to the Internet and that receives access requests, whether under HTTP, FTP or any other conventional or proprietary transfer protocol.

The term "application program," as used herein, encompasses any computer file that contains data in a format for being accessed and processed by the processing unit of a computer.

The term "disk," as used herein, encompasses any storage device that can store computer data and that provides an interface for accessing the stored data.

The term "network," as used herein, encompasses any system comprising a series of computers linked by communications capability and may include the Internet, intranets, telephone networks, wireless and satellite communication networks, or other communications or computer networks.

The term "Internet" means the largest global computer communications network.

The term "World Wide Web" means a large global computer communications network that comprises a significant part of the Internet.

The term "server," as used herein, encompasses any data processing system on which application programs and Internet sites may be stored for access and processing by client computers and encompasses either hardware or software, or a combination thereof, for accomplishing such function.

The term "web browser," as used herein, encompasses any application program which allows for a link to an Internet site and may include programs that

provide for multimedia presentation of information, including text images, sound and video clips.

The term "hypertext link" as used herein, encompasses any graphical icon, button, highlighted text or other symbol that permits a client computer to direct a server to display a page of an Internet site which is associated with the hypertext link.

The term "URL" means "uniform resource locator" and means the address of an Internet site that is accessed by clicking or initiating a hypertext link that is associated with the URL.

The term "HTML" means hypertext markup language, which refers to a language for the creation of pages of Internet sites on the World Wide Web according to the Structured Generalized Markup Language standard.

The term "HTTP" as used herein, shall encompass the "HyperText Transfer Protocol", a protocol under which messages are sent over the Internet from client computers to server computers in the client-server model of distributed computing.

The term "CGI" shall mean "Common Gateway Interface," which shall refer to a specification for communication between a server computer and an application program. Versions of CGI include fast CGI, which permits communication between a server computer and more than one application program running simultaneously.

The term "EDI", or "electronic data interchange" shall mean a protocol for the transfer of data between an application program and a proprietary computer system.

The computer operating systems disclosed herein include a computer operating system that allows a communications device-enabled user to run a plurality of computer applications, independent of the non-communications characteristics of the communications device.

A computer operating system for remote users and methods for providing such a system are provided herein. The operating system may include a user device capable of sending and receiving signals via a communications network, a first server computer having storage, wherein the first server computer includes a processing mechanism for sending and receiving signals via the communications network in selected formats from a set of available communications formats, and a device recognition module of the first server computer for determining the communications format for the user device based on characteristics of the signal from the user device. An application program of the first server may run in response to signals received from the user device according to the selected communications protocol, so that signals from the application program of the first server may be sent by the server signal mechanism to the user device according to the selected communications format. Thus, a remote user may run an application program that resides entirely on a server computer, without the need for local data storage. The absence of a need for local data storage means that a given user can access the operating system from any user device at any location, not just the user's own user device.

The server may include a dedicated location in memory for storing files and other information associated with each user of the operating system. These files may be used to create a unique "desktop" environment for each user, and to store data associated with the user of the same type as a local hard drive or disk would store if the user were using a personal computer. Thus, the dedicated storage creates a virtual hard drive location in memory for the user, the memory being located on the server, rather than on the user's device.

The operating system may further provide a connection to additional servers connected to a network, including the Internet. The operating system may communicate with applications running on these additional servers according to well-known protocols such as HTTP, so that application programs running on the other servers may be accessed and run by the first server, with the results of the applications being sent to the user device and/or stored in the dedicated memory of the server computer for later access by the user. Through use of conventional Internet programming techniques, such as PERL code enabled with fast CGI, application programs running on the first server and on the additional servers may be made to operate independently and to interact, so that applications may send, receive and share data among each other from different locations. The user may also run multiple applications at one time.

The operating system may provide an access control mechanism of the server computer for restricting access to a file stored in the storage that is associated with the user of the user device.

The operating system may be used with any user device that is communications-enabled. Thus the devices may include mainframe computers, desktop personal computers, laptop personal computers, network computers, Internet telephones, pagers, cellular phones, mobile phones, satellite phones, hand-held personal information managers, non-computer appliances, network computers, cable television boxes, web televisions, television sets, and set-top boxes. Communications may be by wireless transmissions, satellite transmissions, fiber optic networks, or other means of transmitting data to the server computer.

Any computing application can be made available under the operating system through the first server or additional servers. Examples include a web authoring application, an electronic mail application, a database application, a search application, a chat application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management application, a drawing application, a presentation application, a translation application, a speech recognition application, and a data formatting application.

The systems and methods described herein provide an operating system that resides solely on the server-side and that permits all computing applications to be accomplished by any device that is capable of a network connection, without the need for local storage for applications. Such applications include Web authoring, Web site management, and communication software technology, including but not limited to full multimedia authoring, online libraries, sounds, forms, e-mail, facsimile, voice-

mail, pager, telephone, financial management, true document printing (as opposed to screen printing), text-to-voice and voice-to-text conversion, file management, spreadsheets, all accessed and run via the Internet. The system resides entirely on an Internet Web Server site and interacts with users via conventional communication protocols. As a result, there is no need for client-side messaging software. All software is provided on the server side. The only software the user needs is any form of Web browser and an electronic communications connection. The system is platform and user device independent. A user may author, create, maintain, send and receive messages from any platform or user device. A user may customize their desktop configuration to run any computing application. For example, a user may switch between applications, transfer text, graphics, or sound files between applications.

In embodiments of the invention, a server computer includes a processing mechanism for receiving signals representing a home page (also known as a Web site or a Web page) from a communications network, converting the signals into a data file, and storing the data file in the memory, a user device adapted for transmitting and receiving signals from the communications network, a communications connection between the server computer and the user device, an access control mechanism connected to the server computer for determining access rights to the data file stored in the memory of the server computer, and a server signal mechanism connected to the server computer and responsive to the access control mechanism, for receiving signals from the user device and for sending signals to the user device, via the communications connection, for generating markup language page signals representative of the data file, wherein the processing mechanism, the access control

mechanism, and the server signal mechanism permit a user of the user device to view, edit, delete, reproduce, or retransmit, or some combination or variation thereof, certain of the data files via interaction with the markup language page signals.

The user device may be selected from the following group of devices, for example: mainframe computers, desktop personal computers, such as, for example, IBM, IBM-compatibles, and MACINTOSH, laptop personal computers, network computers, Internet telephones, pagers, mobile phones, hand-held personal information managers, non-computer (NC) appliances, cable television boxes, television sets, and set-top boxes, or some combination or variation thereof. The user device preferably should include a full or a partial Web browser program, such as, for example, NETSCAPE NAVIGATOR or NETSCAPE COMMUNICATOR, MICROSOFT EXPLORER, MOSIAC, or some combination or variation thereof.

The communications network may be the Internet, may be the World Wide Web, may allow communication via wireless transmissions, or may allow communication via transmissions through fiber optic lines, or some combination or variation thereof, such as, for example, electronic transmissions or radio-wave transmissions. The communications connection may be the Internet or the World Wide Web. The communications connection may allow communication via wireless transmissions, through fiber optic lines, through electronic transmissions, or through some combination or variation thereof. The remote home page authoring system may include a registration mechanism connected to the server computer for storing, accessing, and, optionally, modifying a list of names of registered users, which could

include, for example, individuals, corporations, families, members of particular communities, or shared-interest groups.

The access control mechanism may allow a registered user to create their home page, and to modify, save, reproduce and delete at least a portion of their home page. The access control mechanism may allow a registered user to add text, sound, color, and moving images, or some combination or variation thereof, to their home page. The access control mechanism may allow one or more third parties to view the home pages of one or more registered users, via the communications network, even if the third parties are not themselves registered users.

A method of home page management according to the systems and methods described herein may include providing a server computer for receiving signals representing a home page from a communications network, converting the signals into a data file, and storing the data file, providing a user device for transmitting and receiving signals from the communications network, connecting the server computer and the user device via a communications connection, receiving signals from the user device, sending signals from the server computer to the user device for generating markup language page signals representative of the data file, and determining access rights to the data file, thereby allowing a user of the user device to view the data file via interaction with the markup language page signals if the user is allowed access rights to the data file.

A method of remote home page authoring may further include providing a registration process for allowing users to request registration on the system and for

storing a list of registered users, and, optionally, allowing a registered user of the user device to create their home page and to modify, save, reproduce, or delete, or some combination or variation thereof, at least a portion of their home page. A method of remote home page authoring may further include allowing a registered user of the user device to add text, sound, color, graphics, and moving images, or some combination or variation thereof, to their home page.

Brief Description Of Drawings

FIG. 1 is a schematic diagram illustrating an embodiment of a server-sided Internet based operating system according to the systems and methods described herein.

FIG. 2 is a schematic diagram illustrating the user device that is connected to the system disclosed in FIG. 1.

FIG 3 is a schematic diagram illustrating the server that part of the system depicted in FIG. 1.

FIG. 4 is a flow chart illustrating steps accomplished by the operating system disclosed in FIG. 1.

FIG. 5 is a schematic diagram further illustrating the functional relationship of components of the system disclosed in FIG. 1.

FIG. 6 shows a block outline of the INERGY 2000 operating system. The outline shows the features that may be included in the basic package, as well as add-on functionality that may be added.

FIG. 7 is a schematic illustration of one application that can be accomplished by the operating system of FIG. 1, namely, a web authoring program, and shows that this application may also be connected to the File Manager, the WEBWRITER, and the spell checker, for example.

FIG. 8 is a schematic diagram that illustrates another application that may be run under the operating system of the present invention, namely, a word processing program, and includes examples of some of the editing features that may be available, as well as optional connections to other applications, such as to electronic mail, and to tools such as a file manager and a file format conversion application. Printing, faxing, and connections to other servers also are shown.

FIGs. 9 through 16 show screen shots of screens that are viewed by the user's browser in an embodiment of the invention, including login, folder management, file management, and editing functions.

FIGs. 17 through 60 show examples of Web screen shots and the corresponding source code for such screens, according to embodiments of the systems and methods described herein.

FIGs. 61 through 79 show examples of Web screen shots according to embodiments of the systems and methods described herein. The screen shots reflect functions accomplished by the operating systems of the present invention.

FIGs. 80 through 95 depict source code for embodiments of an operating system of the present invention.

Detailed Description of the Preferred Embodiments

Referring to FIG. 1, the components for an operating system of the present invention are illustrated in a schematic diagram. A user device 20 is connected to a communications network 22. The user device 20 may be any communications-enabled device. In an embodiment of the invention, the user device is a laptop or personal computer with an Internet browser. The only requirement for the user device is that it include communications-enabling capability and some form of user interface. In an embodiment, the user interface is a graphical interface, such as a computer screen or television screen, with an input device such as a keyboard and mouse. The user device 20 does not require any specific software or storage capabilities for running local software applications, other than the minimal amount of software and storage necessary to execute a communications function and to control the user interface. There is no need for client-side messaging software at the user device 20. The user only needs to be communications-enabled, whether by software, by phone, or by other means. To take full advantage of the systems and methods disclosed herein, the user may have a Web browser, such as NETSCAPE NAVIGATOR, NETSCAPE COMMUNICATOR, MICROSOFT EXPLORER, NETCOM, MOSAIC, or any other partial browser, and an Internet, intra-net bulletin board, or other electronic

communications connection or a non-computer communications connection such as one designed for use in connection with a television set, for example WEBTV.

The user device 20 is connected to a communications network 22. The network may be the Internet, the WorldWide Web, or any other communications network, including a fiber optic network, a wireless network, a conventional telephone network, a satellite communications network, or the like. In an embodiment of the invention, the network is a global communications network, such as the Internet. Also connected to the communications network 22 is a server 24. It should be understood that the server 24 could comprise a group of linked servers that accomplish the functions described with respect to the server 24 herein; that is, references to "server" should be understood to include multiple servers, or any other computing device that is capable of performing the functions described herein in connection with the server 24. The server 24 may consist of hardware or server software running on a conventional computer. In an embodiment of the invention, the server is an HTTP server that is enabled for communicating over the Internet in the HTTP communications protocol. The server 24 may be connected to the communications network 22 by any conventional mechanism, such as a network interface card, a modem or a telephone line.

Referring to FIG. 2, the user device 20 is illustrated in schematic format, including an operating system 30 and a communications application 32 operated by the operating system. It should understood that the operating system 30 may be a minimal amount of programming code necessary to enable communications operation and a sufficient user interface to permit the user to interact with the communications

operation. It should be noted that the communications application 32 is the only application necessary for running on the device 20. That is, there are neither memory management nor other applications running on the device 20. It should be understood that the device 20 could include such applications, but such applications are not necessary to accomplish the operating systems disclosed herein. That is, the operating systems disclosed herein operate exclusively on the server side in the conventional client-server model.

Referring to FIG. 3, the server 24 is displayed in a schematic diagram in further detail. An operating system 34 of the server 24 may be a conventional operating system suitable for Internet applications, such as a UNIX operating system. The operating system 34 is coupled with additional code as disclosed herein to provide an extended, server-side operating system. The operating system 34 should be capable of operating a graphical user interface, managing storage, and operating other communications applications, as well providing database management and other data processing, storage and retrieval applications. The operating system controls applications installed on the server 24, including a communications module 38, a device recognition module 42 and other applications 44 that reside in memory 40 of the server 24. The memory 40 may be any conventional storage typical of that included on a server, particularly a server suitable for Internet operations. In an embodiment of the invention, the operating system 34 is a UNIX operating system that is programmed with an Internet programming language, such as PERL. Different applications may be executed simultaneously through use of known Internet programming techniques, such as use of fast CGI scripts, which may be an extension of PERL code. In this embodiment, the UNIX operating system 34 runs the other

applications 44 at all times in a loop. Data from the user device 20, when inputted by the user, is then formatted in the form of a CGI script form that is submitted to the server 24. When the form is submitted, it is recognized by the appropriate programming as calling for processing, which results in the opening of a communications channel, or socket, that calls for processing by the appropriate application 44. Use of PERL code with fast CGI scripts is a well-known technique for permitting multiple applications to be run simultaneously and to be accessed from a local server or a remote server.

Referring to FIG. 4, a flowchart 50 is provided in which steps accomplished by the operating systems of the present invention are illustrated. In a first step 52, a user message is sent by the user device 20 via the network 22 to the server 24. The user message may be any form of signal that is generated by a communications device. In an embodiment of the invention, the user device is a browser, and the user message is an HTTP message; however, the message could be a signal from a cellular phone, a non-computer appliance, a personal information manager, or other device. The signal is not required to be in any particular format, just some type of known format, such as HTTP, fax, phone, or the like; that is, no hardware or software is required in the user device 20 in order to format the signal according to any particular protocol. Thus, any user device 20 that can send a signal can be used in connection with the operating systems disclosed herein. Because the system is platform and operating system independent, a user may author, create, maintain, send and receive messages from any platform, including but not limited to IBM PC and compatible platforms, MACINTOSH platforms, and non-computer (NC) or set-top boxes such as WEBTV, using any operating system, including but not limited to MICROSOFT WINDOWS,

WINDOWS 95, WINDOWS NT, WINDOWS CE, DOS, UNIX and proprietary or legacy operating systems.

At a step 54, in response to any incoming signal, the operating system 34 of the server 24 initiates the device recognition module 42. The device recognition module provides a function that may be written in an conventional programming language, such as PERL. The device recognition module 42 executes a matching algorithm that compares the characteristics of the signal to a set of known signal characteristics. In some cases, recognition may be accomplished in part by recognizing the input channel of the signal; thus, a signal coming in over a telephone line may be recognized as either a voice signal, a telephone keypad signal, or a modem signal by virtue of the fact that it is sent over a telephone line. Similarly, if the signal comes through the network interface card of the server 24, then the device is recognized as being a device capable of sending HTTP protocol data, such as a browser. Known user devices have characteristics that are unique and identifiable when a signal from the user device 20 is sent over the communications network 22 to the server 34. A truth table can be generated that associates each type of known user device, such as each browser, with the signal characteristics and the capabilities of that browser. The table may created by a standard matching algorithm written in a conventional language such as PERL. In the case of many browsers, the signal includes the device name and type in the message codes. The signal characteristics are matched against a known library of such signal characteristics, or simply the device name, permitting the device recognition module 44 to identify the nature of the user device 20. Identification of the user device 20 is thus accomplished without need for special user input, avoiding the need for software at the user device 20 that formats

signals according to a particular protocol or that identifies the user device 20 as being of a certain type.

Next, at a step 58, the operating system 34 of the server 24 may initiate the communications module 38. The communications module 38 has a number of features that accomplish the sending and receiving of data. First, the communications module 38 is capable of receiving data in any of a wide variety of known formats and converting the data to an appropriate format for storage, manipulation, and retrieval. The communications module 38 processes data in the format that is appropriate for the user device 20 that was identified by the device recognition module 42 at the step 54. For example, if the user device 20 is identified as an Internet browser, the communications module 38 recognizes that data should be processed in HTTP format. If the user device 20 is identified as a telephone, then the communications module 38 may initiate voice recognition code that converts voice signals from the user device 20 into text, which can then be used for further processing. It should be understood that the communications module 38 may consist of a suite of different applications that are running on the server 24. Some of the applications that comprise the communications module 38 may reside on the third party server 28. The operating system 34 controls the communications module 38, which initiates all of the functions necessary to receive and properly format signals for further processing. Applications or code that accomplish reformatting of data signals are well-known. Such applications include voice recognition software, such as Dragon Systems' Naturally Speaking software, which converts from voice to text and from text to voice, as well as software that converts files between different text and word-processing formats, between text and

graphics formats, and the like. The communications module 38 preferably includes a suite of applications that can convert data between any two desired formats.

In addition to receiving signals and converting them into the proper format for processing, the communications module 38 permits the server 24 to format a signal to be sent over the communications network 22 to the user device 20. The format of the signal 22 is determined by the communications software 38, based upon the type of user device 20 detected at the step 54. Thus, if the device is a personal computer with a browser, then the communications software 38 may send a signal that is according to the Internet HTTP protocol. Based on the characteristics of the particular user device 20, such as a particular browser, the signal may be formatted to conform to the characteristics of that browser, which are identified in the truth table that is included in the device recognition module 42. If the device is a phone, the communications module 38 may send a voice signal, which has been generated by voice generation software code. The content of the signal sent to the user device 20 will be based on the processing selected by the user. The user sends signals via the user device 20, the signals are processed by the communications software 38, and the signals are then relayed in proper format for further processing as described below. As processing of signals is completed, the results are sent back to the user device 20 in the proper format, so that the user can obtain the results of computer processing. Thus, the communications software 38 permits the user to interact with the server 24 in a variety of data processing formats, from any user device in any location, without the need for special software for communicating with the server 24.

Once a communications link has been established at the step 58 between the server 24 and user device 20, thus formatting any incoming signal in an appropriate format for further processing, the user may interact with the other software applications 44 running on the server 24. In an embodiment, as depicted in the flowchart 50 at a step 60, a first application is a log-in application. The log-in application may be a conventional log-in application, such as a table with entries to be completed by the user. New users may be prompted to enter information that will be stored in the memory 40 of the server 24. In order to receive an account on the system, a user may be required to register, e.g., through their television, mail, telephone, facsimile, or on-line. Existing users may be prompted to enter a password, which will be compared to the password for the user stored in the memory 40. The log- in application may be any conventional type of log-in application, and may be written in a conventional, Internet-based language, such as Java.

In an embodiment in which the user device is a browser, when a previously-registered user logs in to the operating system 10, the user's login ID and password are sent, using HTTP, to the server 24, which may be located anywhere, and then passed, using CGI script code, to the operating system 10, which then accesses the user's profile from the database of user profiles, using conventional database software, such as Berkeley database software. Based on the data in the user's customer profile, a set of scripts are run, which then are sent back to the Web server, which passes the HTML instructions to the user's browser, using HTTP, and the user's browser then displays the user's customized desktop configuration, as discussed below.

Once the user logged in, at a step 62 the operating system 34 of the server 24 may call memory management software that is running on the system for controlling the data storage memory 40 of the server 24. The memory management software may then store information obtained at the log-in step 60 in a location in the memory 62. The memory management software may be used to partition a location in memory that is dedicated to the particular user. The partitioned location is thus a "virtual" hard drive, mimicking the functions of a conventional disk, hard drive, or other storage device that would normally reside on a personal computer on the client side, i.e., on the user device 20. The presence of the virtual hard drive in the memory 40 avoids the need for storage at the user device 20. The memory management software runs throughout the operation of operating systems disclosed herein, so that operation of any applications 44 results in storage, retrieval, and manipulation of information in storage in a conventional manner similar to that which is performed by other memory management applications for conventional client personal computers. In an embodiment of the invention, memory management software consists in part of Berkeley database software.

The memory 40 may include files associated with the particular user. Referring to FIG. 5, a schematic diagram illustrates a method of implementation of a server-sided Internet based operating system according to the present invention. Once on the web site at the log-in step 60 of FIG. 4, a new user may register for access to one or more of the applications 44 on the operating system's suite of desktop applications 44, including such programs as a word processor, e-mail, chat, web-authoring software, a contact manager, a personal information manager ("PIM"), spreadsheet software, voice-to-text & text-to-voice software, financial management

software (including electronic checkbook and bill payment), personal and corporate bookkeeping, data management, desktop publishing, desktop (or WEBTOP) administration, meeting manager/scheduler, etc., all of which may reside on the server 24. These applications 44 may be written in any programming language, such as JAVA, PERL, CGI scripts, C, C++, LISP, or the like. In an embodiment, an Internet programming language, such as JAVA or HTML is used. These applications 44 may also be conventional off-the-shelf applications that are acquired from third parties and installed on the server 24 or a third party server 28, programmed by conventional means. These applications may be interlaced with online creative multimedia tools. At the log-in step 60, the new user may then be given a password, which they may modify, and is given a standard desktop configuration of the applications, including a standard background.

Referring to FIG. 5, a database, which may reside on the server 24 (which may be a server farm or on some other server), may be maintained that includes a customer profile for each customer. The customer profile, which may include one or more customized desktop configurations, may be updated by the user either consciously, through selection of an option such as "SETUP", or may be automatically updated as the user rearranges or otherwise modifies their desktop while they are actively connected to the operating system. Each user may customize their desktop arrangement. For example, users may upload graphics, text, or sounds to use as a background. Users may add and remove hypertext links to their favorite Web sites. Users may modify the size, shape or arrangement of the icons or symbols that permit access to each of the applications, etc. A user may have one customized desktop configuration for use in the office and another customized desktop configuration for

use at home. In addition, a family may have a common desktop configuration (or set of configurations), but different passwords for e-mail for each of the family members, so that each of the family members can have private e-mail. In other words, the files stored in the memory 40 of the server 24 dedicated to the user may include a file associated with the selected configuration of the interface that the user desires. Thus, upon logging in at the step 60 and initiating the memory management software 62, the system 10 may, at a step 64, display the "desktop" for the user that permits the user to select an application.

The user may customize the desktop by interacting with an application that permits the user to select applications that the user wishes to make available. It should be understood that different "desktops", or initial files, may be stored for each user, corresponding to different input devices. For example, if the user device 20 is recognized to be a hand-held computer without a keyboard, then a "desktop" file may be created that includes a keyboard that can be used by pointing and clicking with a mouse. If the user device is recognized to be a voice telephone, the "desktop" may consist of a series of voice signals that invite the user to select a particular command from a menu of commands. For example, the voice command could state: "to send or hear e-mail, say 'e-mail'." Thus, a file customized to the user creates a unique desk top appearance (or sound) that is sent by the communications module 38 to the user device 20 over the communications network 22. In an embodiment, the user device is a personal computer, and the graphical user interface displays a page, such as an HTML page, that depicts icons as graphical objects that can be clicked by the user using a mouse or cursor to select a particular application. In a different embodiment, if the user device 20 is a cellular phone and the signal is a keypad tone from the

phone, then the user "desktop" file for a cellular phone may be loaded, so that the user may select an application by pressing a number on the keypad of the phone.

In an embodiment, once the "desktop" file is displayed, the user can then access any of the applications for which they have registered, which may include one or more of the applications shown in the "desktop" layer of the diagram in FIG. 5. Each of the desktop applications also may have access to one or more of the tools in the "TOOLS" layer of the diagram in FIG. 5. For example, a user may receive an e-mail message with an attached document in MICROSOFT WORD format. The user may wish to edit the document using the WEBWRITER program. The File Manager tool will allow the user to convert the MICROSOFT WORD document into Text format, which the WEBWRITER can read. The user can then edit the document and user File Manager to convert the revised document back to MICROSOFT WORD format and send it back as an attachment to an e-mail reply message. Or, the user could convert the document for use in the spreadsheet or bookkeeping application. The user also may even add the document to their personal or business Web site. A variety of additional options will be apparent. For example, the spell-check tool may be accessed from the word processing program, as well as from the e-mail program or the Web site creation and management program, etc.

The "desktop" file preferably includes a menu from which the user may select other applications. It may also include news feeds, bookmarks, advertising, and other features. That is, the desktop file should be understood to be similar to the main menu that appears after logging on to a personal computer. Using icons, text, voice, or other prompts, the desktop makes available for the user other applications 44. The user may

thus select a particular application through the input of the user device 20. The other applications 44 may be any computing applications. The user may add or delete applications from the menu that appears on the "desktop" file by interacting with the file in a conventional manner. For example, a pull-down menu may exist at the top of the screen that permits the user to "add" or "delete" an item that has been high-lighted by clicking a mouse.

Referring to the flowchart 50 of FIG. 4, at a step 64 the user selects an application. Having selected an application, the operating system determines whether the application is stored locally or on a third party server 28 at a step 68. A local application is an application running on the server 24. If the local application is selected, then the operating system 34 of the server 24, which is running all of the applications 44 in a loop, will result in the appropriate application recognizing a form sent by CGI script that is recognized by the application as calling for processing. Having recognized the form, the application initiates processing in a step 70 and runs the application. Signals based on the application may be sent by the server 24 over the network 22 to the device 20. Thus, at the user device 20, pages are displayed, voice signals are heard, or other signals are processed based on the results of the application. If the application is an application that produces output in a particular format, such as text, then the communications module 38 re-formats the output into a format appropriate for the user device 20. Thus, if the user device is a phone, and the output of the application is text, then the communications module 38 initiates voice-generation software (which may reside on the server 24 or on a third party server 28) that converts the text to voice and plays the voice message to the phone. Thus, a user can use any application 44 as if the user were using an application located in storage

of the user device 20, even though that device may have only sufficient storage to enable communications and a user interface.

Upon completion of the application selected at the step 64, or at any other time, the user may initiate another application at a step 72. If the user wishes to run another application, then the user is returned to a menu at the step 64. If at the step 72 the user does not wish to execute another application, then the user may end the program at a step 74 by any conventional means, including turning off the device or logging out.

If at the step 68 the application is a third party application, rather than a local application, then the server 24 initiates the software that establishes a communications link at a step 78 to another server. The other server may be the third party server 28 depicted in FIG. 1. The communications link may be established by the communications module 38. Thus, the third party device could be a server or other computing device that is capable of communicating over a communications network, such as the communications network 22. Once the communications link is established, the server 24 may send signals to the third party server 28 which is running other applications 44 in a loop, under an operating system such as a UNIX system. Signals may be sent using PERL code with fast CGI script capability. The user signal is formatted to initiate the application, such as a CGI script form, which results in recognition by the selected application 44 running on the third party server 28. While the application is running, the server 24 receives signals reflecting the processing of the third party 28 and relays those signals over the communications network 22 to the user device 20 in the proper format, as described above. Once the

application is complete, or during running of an application, the user may at a step 72 execute another application, at which point the user may select an application at menu at the step 64. Alternatively, if processing is complete, then the user may end at the step 74 in a conventional manner. Switching between running applications can be accomplished by the operating system in a conventional Internet programming manner, such as through use of PERL code enabled with fast CGI scripts.

Thus, embodiments of the invention permit the user to accomplish any computing function through use of an operating system that does not require the user device 20 to have storage or to have installed software applications, other than minimal amounts required to enable communications. Thus, embodiments include web authoring software, contact management software, search and retrieve software, translation software, image retrieval and manipulation software, electronic mail software, and data manipulation and formatting software. These software applications may be made compatible with each other so that data can be transferred among them to create a suite of interconnected software applications, all running under an operating system residing solely on the server 24, or on a third party server 28 that is accessible over the communications network 22 by the server 24. Thus, the system 10 appears and acts in all respects to the user like an operating system installed in the storage of a personal computer, and it accomplishes all computing functions normally accomplished by a mainframe or personal computer. However, the user is not required to use a particular user device, and the user device is not required to have more than minimal hardware or storage. Moreover, the same user may use different devices at different locations to accomplish computing functions, since all of the

user's files are stored in virtual memory at the server 24. Certain embodiments and applications of the operating systems disclosed herein are further disclosed below.

Applications that can be accomplished by the systems and methods described herein include Web authoring, Web site management, communications, full multimedia authoring, online libraries, sounds, forms, e-mail, chat, facsimile, voice-mail, pager, telephone, financial management, true document printing (as opposed to screen printing), text-to-voice and voice-to-text conversion, file management, and spreadsheets, all accessed and run via the Internet. A user may also create, receive, edit, and print documents (in which case the user would require a local printer attached to the user device 20, or would be required to wait for a document to be printed by a printer attached to the server 24 and otherwise transmitted to the user), run financial packages, pay bills electronically, convert text-to-voice and voice-to-text, manage personal information, schedule appointments, run desktop publishing software, send and receive facsimile messages, etc.

In an embodiment of the invention, the system resides entirely on an Internet Web Server site and interacts with users via standard hyper-text markup language ("HTML"), which is described in Ian S. Graham, HTML Sourcebook, 3d ed. (1997), PERL, CGI, and JAVASCRIPT programs, which are written for a universal protocol, currently HTML Standards Version 3.0. The system also includes code written in UNIX, PERL, JAVA, and C++. General techniques of Internet programming are described in Kris Jamsa, Ph.D. and Ken Cope, Internet Programming (1995). Data storage, manipulation and retrieval functions may be accomplished by conventional database programs, such as Berkeley, Oracle, or similar programs.

FIG. 6 shows a block outline of the INERGY 2000 operating system. The outline shows the features that may be included in the basic package, as well as add-on functionality that may be added.

FIG. 7 is a schematic illustration of one application that can be accomplished by the operating system of FIG. 1, namely, a web authoring program, and shows that this application may also be connected to the File Manager, the WEBWRITER, and the spell checker, for example.

FIG. 8 is a schematic diagram that illustrates another application that may be run under the operating system of the present invention, namely, a word processing program, and includes examples of some of the editing features that may be available, as well as optional connections to other applications, such as to electronic mail, and to tools such as a file manager and a file format conversion application. Printing, faxing, and connections to other servers also are shown.

FIGs. 9 through 16 show screen shots of screens that are viewed by the user's browser in an embodiment of the invention, including login, folder management, file management, and editing functions.

FIGs. 17 through 60 show examples of Web screen shots and the corresponding source code for such screens, according to embodiments of the systems and methods described herein.

FIGs. 61 through 79 show examples of Web screen shots according to embodiments of the systems and methods described herein. The screen shots reflect functions accomplished by the operating systems of the present invention.

FIGs. 80 through 95 depict source code for embodiments of an operating system of the present invention.

Another advantage of the operating system disclosed herein is that no particular Internet Service Provider (ISP) is required. A user may continue to receive messages at the same address (located on the server 24), with no interruption, even if the user switches ISPs, technologies, computer terminals, or televisions, or if the user moved to a different location. The user would not lose messages if the user's laptop or portable computer were stolen, because the messages would reside on the system's server and would be backed-up automatically.

Another advantage for the user would be that, because all system software is provided directly at the system's Web site, all enhancements are provided at a centralized location rather than having to periodically provide each user with updated software. The user does not need to upgrade software.

The system may include additional features such as the ability to convert files received from many types of systems to HTML and TXT, support for embedded application files that includes automatically executing the corresponding application, an address book that automatically addresses a message to send when an address entry is selected, and general access to a list of registered system users. the PIM feature

provides the ability to create and store for global retrieval a complete listing of the user's contacts. A "schedule" feature may offer the ability to create and maintain a complete 24-hour schedule of the users activities including comprehensive connectivity to their PIM, for global retrieval. A user, including a user with only a set-top box or an Internet phone or pager, can send a document to print. The document will go to a print server, and will then be spooled in a data stream over the Internet to the printer designated by the user. Thus, true document printing, as opposed to screen capture, may be provided.

Other features of the operating system include the ability to interface e-mail communications with facsimile, voice and multimedia communications. A preferred feature of the operating system may be to connect all of the applications to each other, which would allow users to access all of the other applications from within a particular application.

The operating system also may have security features, including but not limited to support for digital signatures, encryption, and password protection, as well as a time out feature to prevent access to the user's information if the user has not exited the service or taken any other action for an extended period of time.

The system also may have notification via telephone, facsimile, pager, or other device when e-mail has been received. The system may have additional gateways built to offer the seamless carry over to existing corporate mail and messaging systems such as CCMAIL.

One of many possible ways to implement they system would be to use clustered DEC 64-bit alpha servers (or a server farm) to allow for safe fail-over, distribution of operating load, and scalability. A UNIX operating system could be used. Database programs, such as Berkeley or Oracle databases, and Webserver programs, such as Apache or Netscape Webserver programs, could run on the servers, and the servers could be connected to the Internet through T1 or T3 lines, or other communication channels with wide-band capability.

All patents, patent applications, articles, books and other references cited herein are incorporated herein by reference.

While the invention has been disclosed in connection with the preferred embodiments shown and described in detail, various modifications and improvements will be apparent to one of ordinary skill in the art from the above description, including, for example, continuously updating the customer profile, or updating the customer profile at predetermined intervals or points, such as when a user exits the system.

We claim:

1. A computer operating system for a remote user of a user device capable of sending and receiving signals via a communications network, comprising:

a first server having memory, wherein the first server includes a communications module for sending and receiving signals via the communications network in one or more selected formats from a set of available formats; and

a device recognition module of the first server for determining the type of user device, so that the communications module sends and receives signals in a format selected for the user device.

2. A computer operating system according to claim 1, further comprising:

an application program of the first server that runs in response to signals received from the user device according to the selected communications format.

3. A computer operating system according to claim 1, further comprising a location in the memory of the server for storing information associated with the user of the user device.

4. A computer operating system according to claim 3, wherein the information includes files associated with application programs selected by the user of the user device.

5. A computer operating system according to claim 4, further comprising:

a second server connected to the communications network; and

an application program of the second server, wherein the second server executes the application program of the second server according to a signal from the first server that is formatted by the communications module.

6. A computer operating system, according to claim 1, further comprising:

an access control mechanism of the server for restricting access to a file stored in the memory that is associated with the user of the user device.

7. The computer operating system of claim 1, wherein the user device is selected from the group consisting of: mainframe computers, desktop personal computers, laptop personal computers, network computers, Internet telephones, pagers, cellular phones, mobile phones, satellite phones, hand-held personal information managers, non-computer (NC) appliances, cable television boxes, web televisions, television sets, and set-top boxes.

8. A computer operating system, according to claim 1, wherein the communications network allows communication via wireless transmissions.

9. The computer system of claim 2, wherein the application program of the first server is selected from the group consisting of a web authoring application, an electronic mail application, a database application, a search application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management

application, a drawing application, a presentation application, a translation application, a speech recognition module, and a data formatting application.

10. The computer system of claim 5, wherein at least one of the application program of the first server and the application program of the second server is selected from the group consisting of a web authoring application, an electronic mail application, a database application, a search application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management application, a drawing application, a presentation application, a translation application, a speech recognition module, and a data formatting application.

11. A method of providing a computer operating system, comprising:

providing a user device capable of sending and receiving signals via a communications network according to a communications format;

providing a first server having memory, wherein the first server includes a processing mechanism for sending and receiving signals via the communications network according to a selected communications format from a set of available communications formats;

determining the communications format for the user device based on characteristics of the signal from the user device;

receiving signals from the user device; and

sending signals to the user device according to the selected communications format.

12. A method of providing a computer operating system according to claim 11, further comprising:

providing an application program of the first server;

running the application program in response to signals received from the user device according to the selected communications format; and

sending signals from the application program to the user device according to the selected communications format.

13. A method of providing a computer operating system according to claim 12, further comprising:

storing information associated with the user of the user device in the memory of the first server.

14. A method of providing a computer operating system according to claim 13, wherein the information includes files associated with application programs selected by the user of the user device.

15. A method of providing a computer operating system according to claim 14, further comprising:

providing a second server connected to the communications network;

providing an application program of the second server;

sending a signal from the first server to the second server according to input from the user device;

running the application program of the second server according to the signal from the first server;

sending a signal corresponding to the results of the application program of the second server to the first server; and

sending the results of the application program of the second server to the user device according to the selected communications format.

16. A method of providing a computer operating system according to claim 15, further comprising:

storing the results of the application program of the second server in the file associated with the user in the memory of the second server.

17. A method of providing a computer operating system, according to claim 11, further comprising:

restricting access by a user to a file stored in the memory that is associated with the user of the user device.

18. The method of providing a computer operating system of claim 11, wherein the user device is selected from the group consisting of: mainframe computers, desktop personal computers, laptop personal computers, network computers, Internet telephones, pagers, cellular phones, mobile phones, satellite phones, hand-held personal information managers, non-computer (NC) appliances, cable television boxes, web televisions, television sets, and set-top boxes.

19. The method of providing a computer operating system of claim 11, wherein the communications network allows communication via wireless transmissions.

20. The method of providing a computer operating system of according to claim 12, wherein the application program of the first server is selected from the group consisting of a web authoring application, an electronic mail application, a database application, a search application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management application, a drawing application, a presentation application, a translation application, a speech recognition module, and a data formatting application.

21. The method of providing a computer operating system of according to claim 12, wherein at least one of the application program of the first server and the application program of the second server is selected from the group consisting of a web authoring application, an electronic mail application, a database application, a search application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management application, a drawing application, a presentation application, a translation application, a speech recognition module, and a data formatting application.

22. A computer operating system that allows a communications device-enabled user to run a plurality of computer applications, independent of the non-communications characteristics of the communications device.

23. The computer operating system of claim 22, wherein, the user may switch between applications.

24. The computer operating system of claim 22, wherein the user may process data between applications.

25. The computer operating system of claim 22, wherein the computer applications are selected from the group consisting of: a web authoring application, an electronic mail application, a database application, a search application, a graphics application, a personal information manager applications, a scheduler application, a calendar application, a word processing application, a spreadsheet application, a calculator application, a document management application, a drawing application, a presentation application, a translation application, a speech recognition module, and a data formatting application.

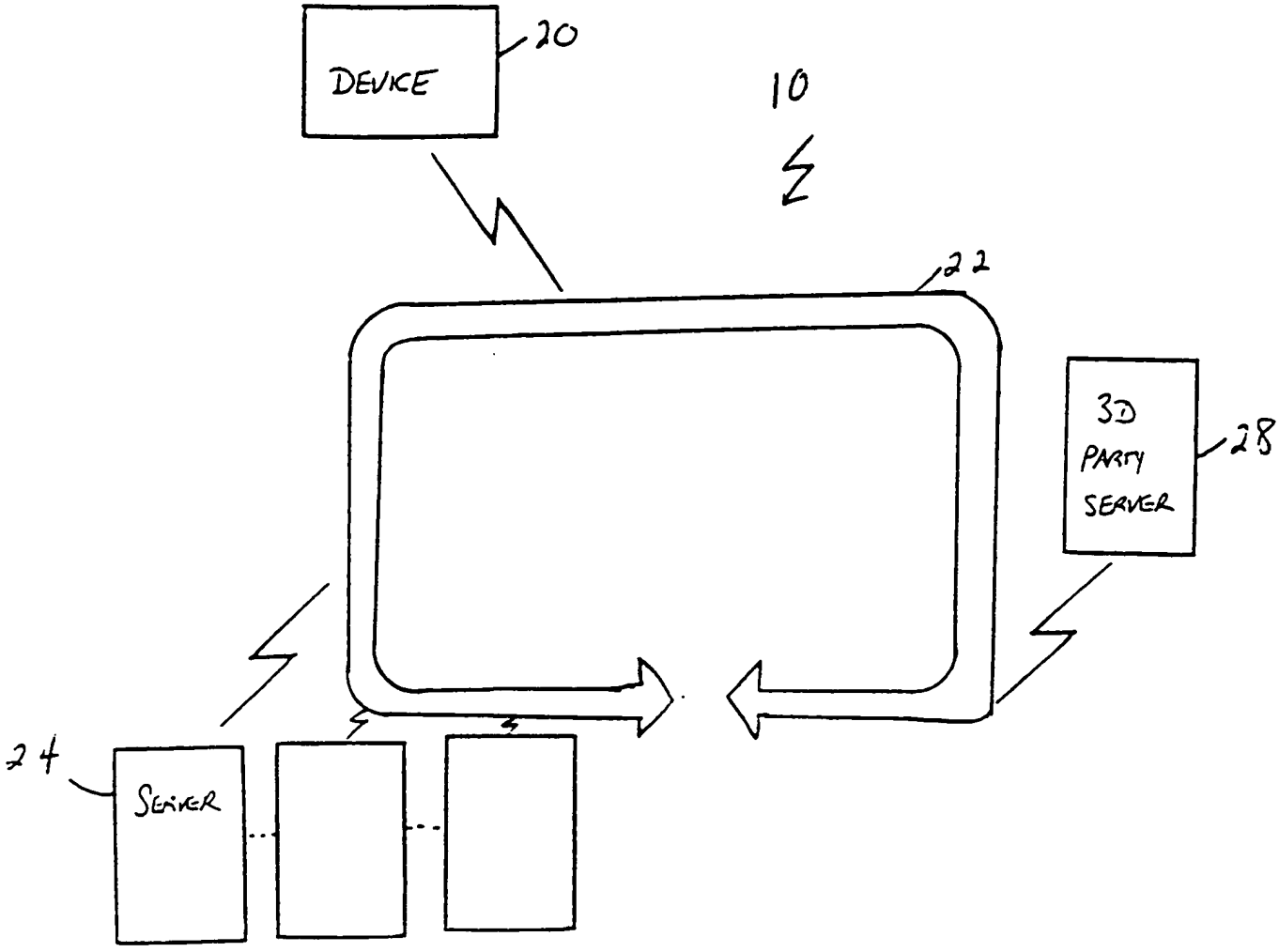


FIG. 1

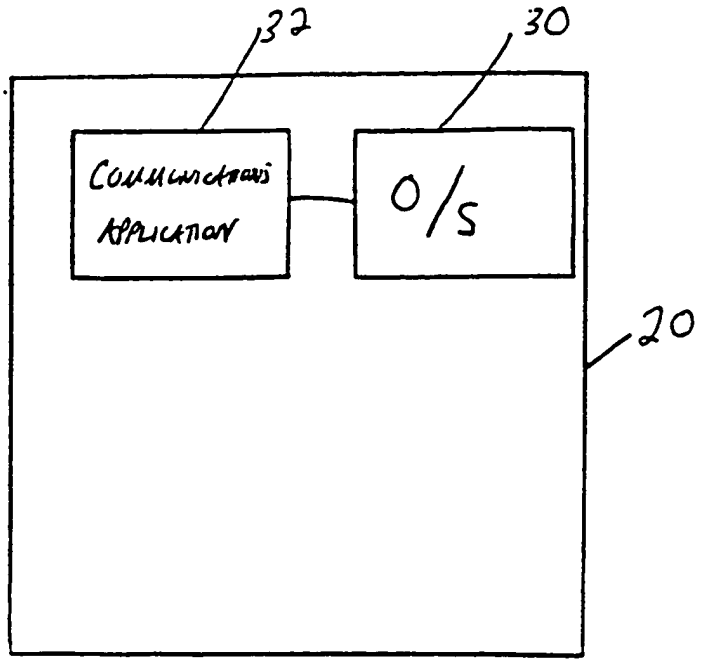

32
30
COMMUNICATIONS APPLICATION
O/S
20

FIG. 2

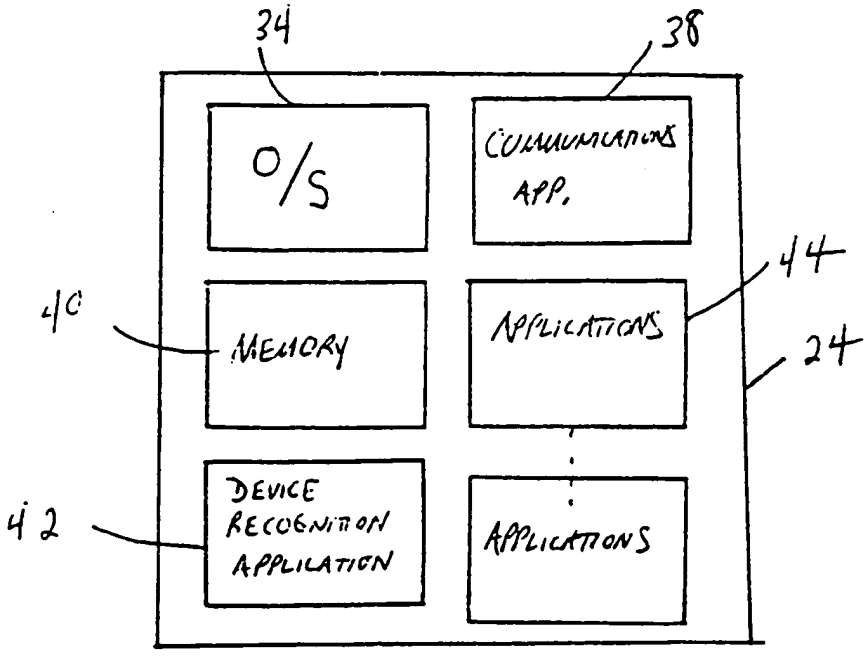

34
38
O/S
COMMUNICATIONS APP.
44
40
MEMORY
APPLICATIONS
24
DEVICE RECOGNITION APPLICATION
42
APPLICATIONS

FIG. 3

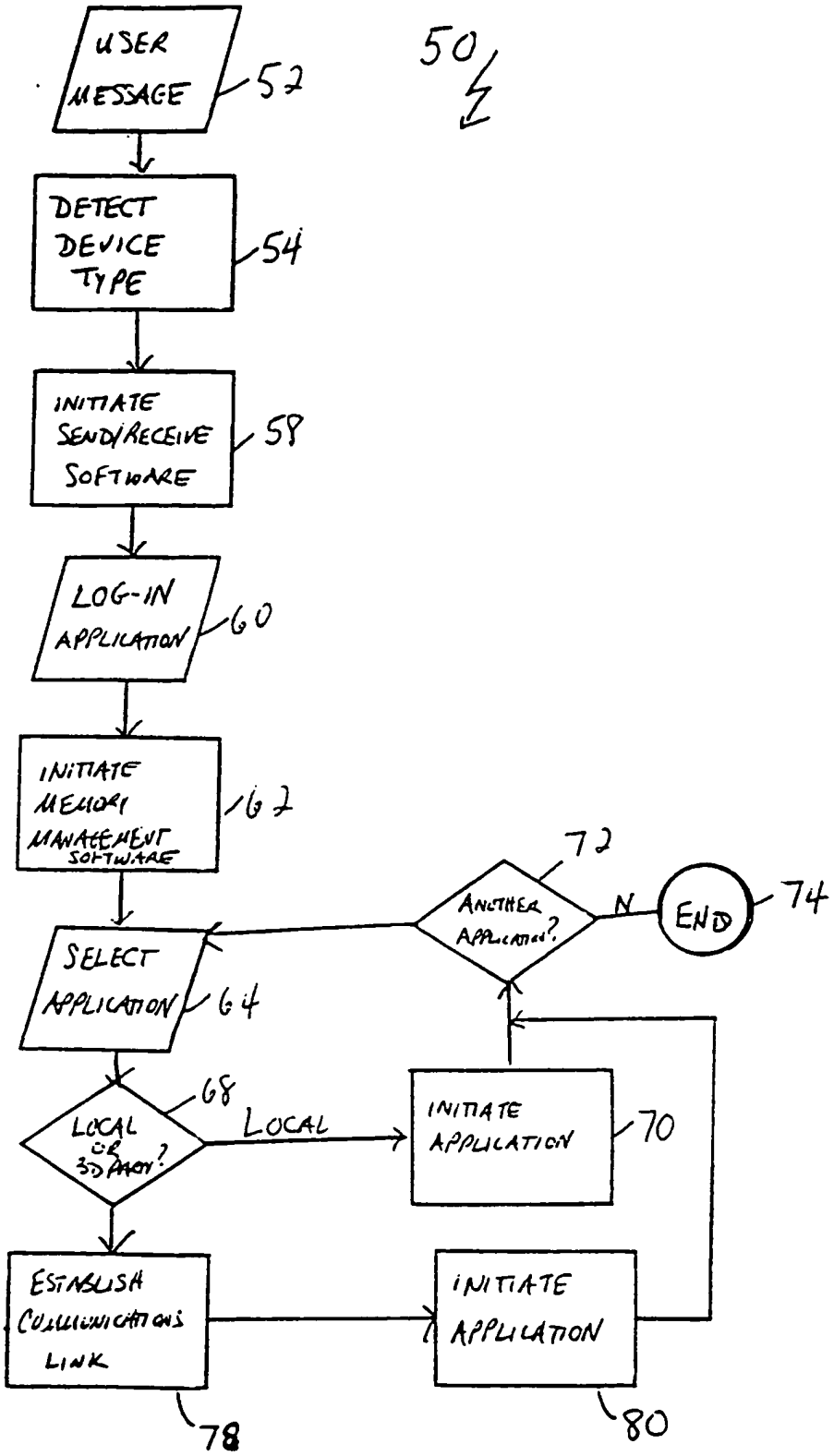

50
USER MESSAGE
52
DETECT DEVICE TYPE
54
INITIATE SEND/RECEIVE SOFTWARE
58
LOG-IN APPLICATION
60
INITIATE MEMORY MANAGEMENT SOFTWARE
62
SELECT APPLICATION
64
LOCAL OR 3D PARTY?
68
LOCAL
INITIATE APPLICATION
70
ANOTHER APPLICATION?
72
N
END
74
ESTABLISH COMMUNICATIONS LINK
78
INITIATE APPLICATION
80

FIG. 4

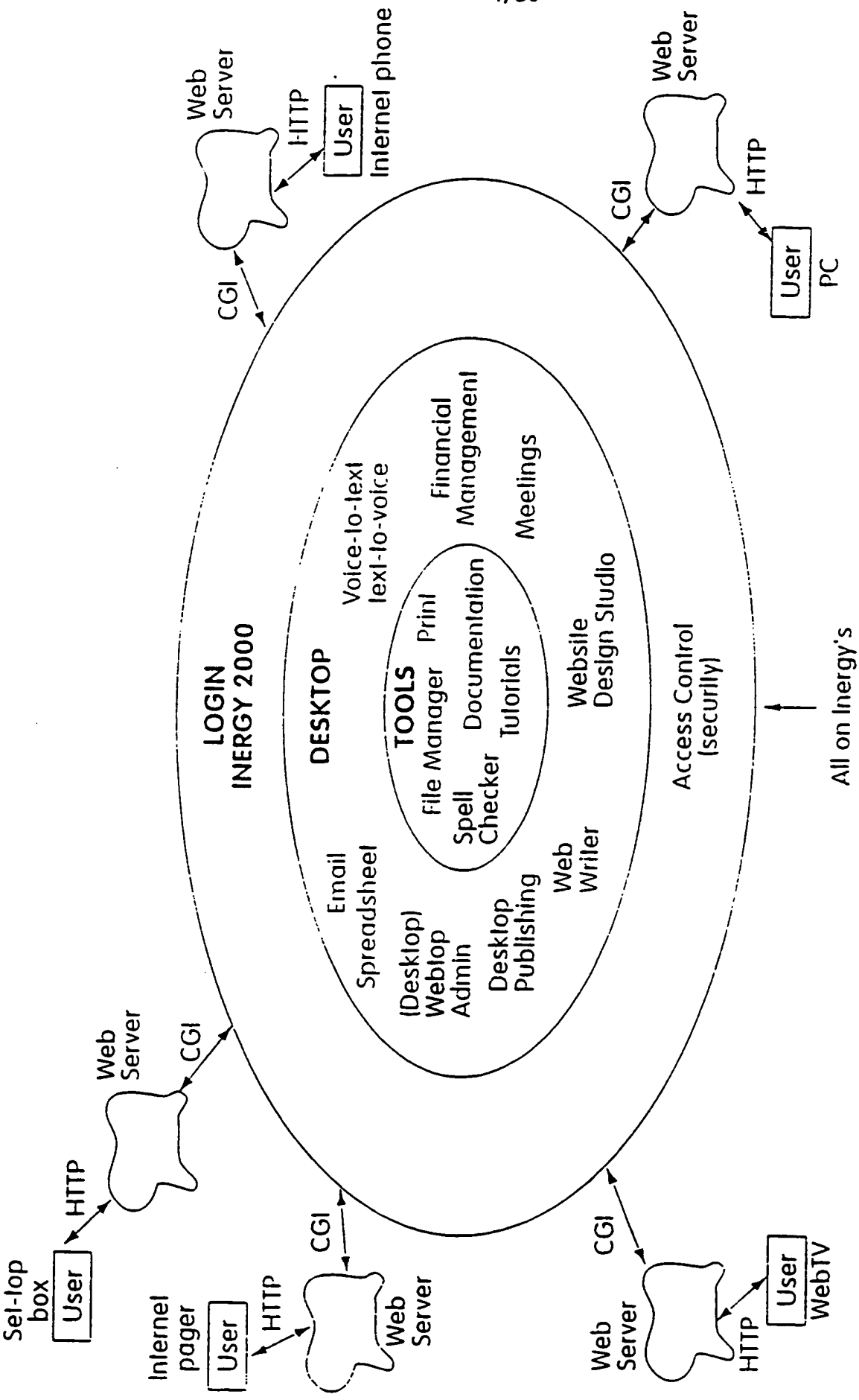

Web Server
HTTP
User
Internet phone
CGI
Web Server
CGI
HTTP
User
PC
LOGIN
INERGY 2000
DESKTOP
Voice-to-text
text-to-voice
Financial Management
Meetings
TOOLS
Print
File Manager
Documentation
Spell Checker
Tutorials
Website Design Studio
Email
Spreadsheet
(Desktop)
Webtop
Admin
Desktop Publishing
Web Writer
Access Control
(securityl)
All on Inergy's
DEC Sever Farm
Sel-top box
User
HTTP
Web Server
CGI
Internet pager
User
HTTP
CGI
Web Server
Web Server
CGI
HTTP
User
WebTV

FIG. 5

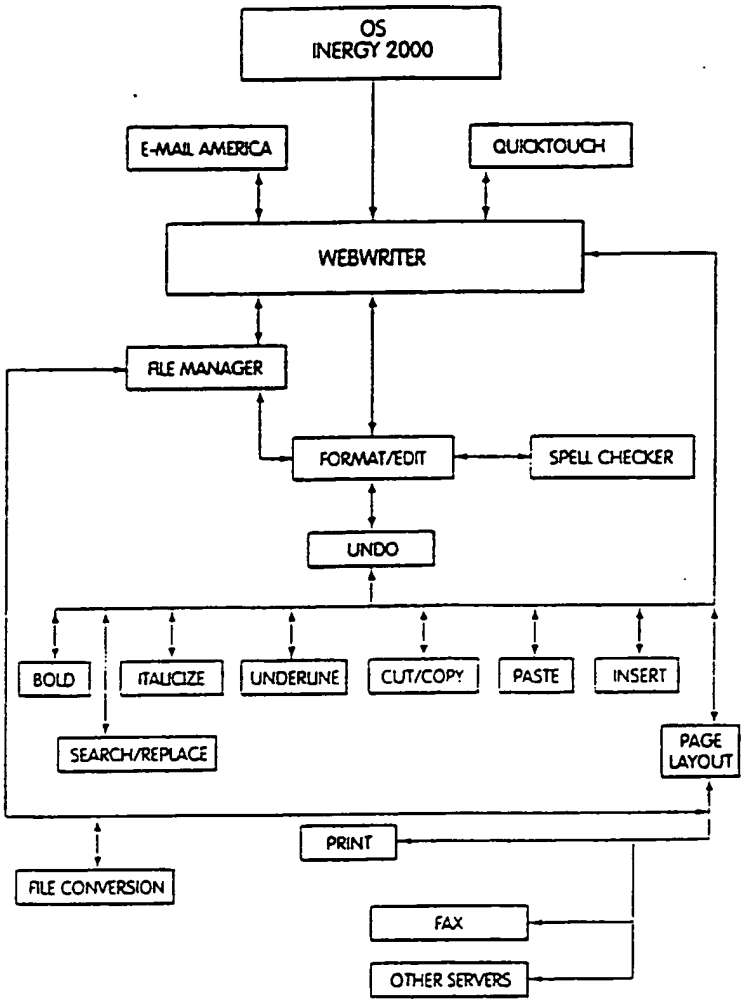

OS
INERGY 2000
E-MAIL AMERICA
QUICKTOUCH
WEBWRITER
FILE MANAGER
FORMAT/EDIT
SPELL CHECKER
UNDO
BOLD
ITALICIZE
UNDERLINE
CUT/COPY
PASTE
INSERT
SEARCH/REPLACE
PAGE LAYOUT
PRINT
FILE CONVERSION
FAX
OTHER SERVERS

FIG. 6

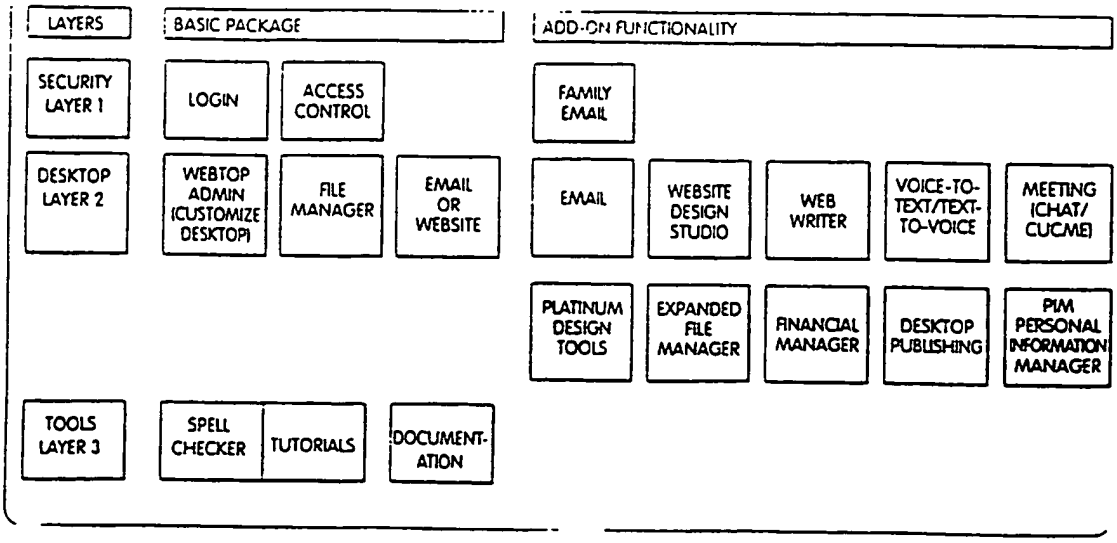

LAYERS
BASIC PACKAGE
ADD-ON FUNCTIONALITY
SECURITY LAYER 1
LOGIN
ACCESS CONTROL
FAMILY EMAIL
DESKTOP LAYER 2
WEBTOP ADMIN (CUSTOMIZE DESKTOP)
FILE MANAGER
EMAIL OR WEBSITE
EMAIL
WEBSITE DESIGN STUDIO
WEB WRITER
VOICE-TO-TEXT/TEXT-TO-VOICE
MEETING (CHAT/ CUCME)
PLATINUM DESIGN TOOLS
EXPANDED FILE MANAGER
FINANCIAL MANAGER
DESKTOP PUBLISHING
PIM PERSONAL INFORMATION MANAGER
TOOLS LAYER 3
SPELL CHECKER
TUTORIALS
DOCUMENT-ATION

FIG. 7

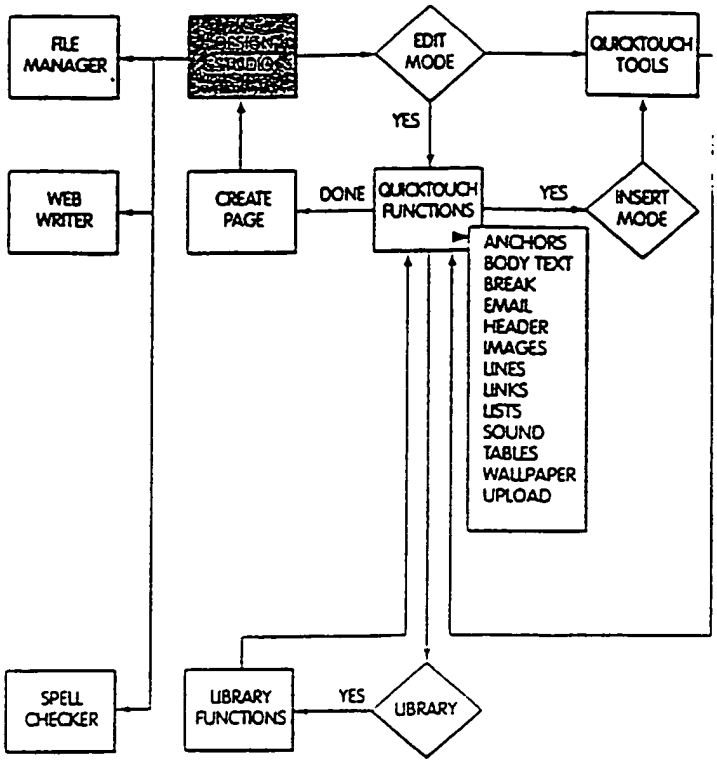

FILE MANAGER
EDIT MODE
QUICKTOUCH TOOLS
YES
WEB WRITER
CREATE PAGE
DONE
QUICKTOUCH FUNCTIONS
YES
INSERT MODE
ANCHORS
BODY TEXT
BREAK
EMAIL
HEADER
IMAGES
LINES
LINKS
LISTS
SOUND
TABLES
WALLPAPER
UPLOAD
SPELL CHECKER
LIBRARY FUNCTIONS
YES
LIBRARY

FIG. 8

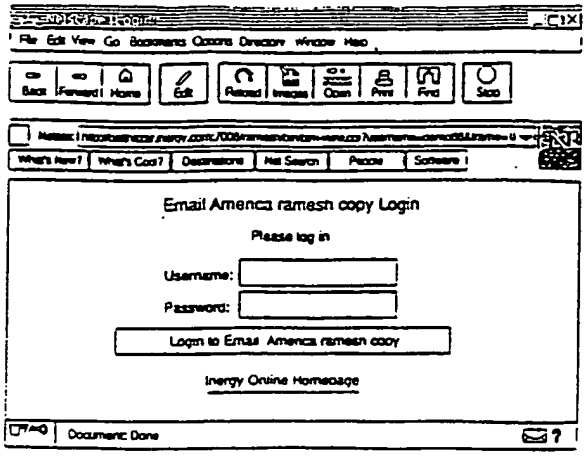


FIG. 9

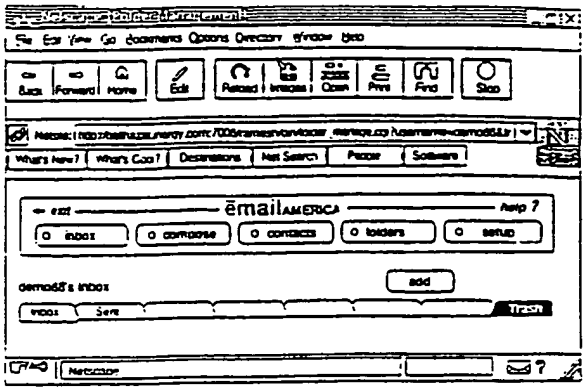


FIG. 10

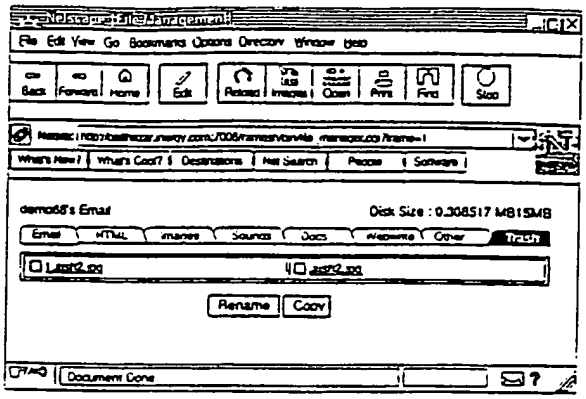


FIG. 11

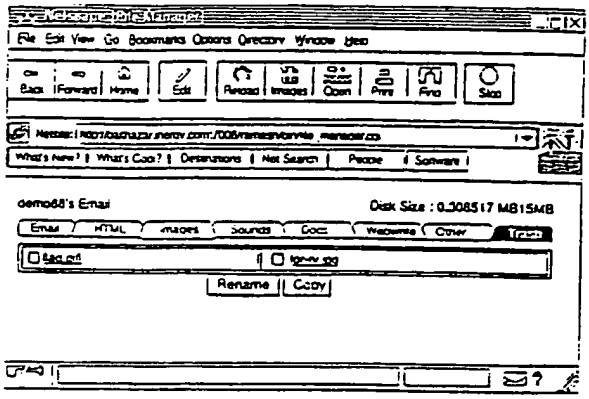


FIG. 12

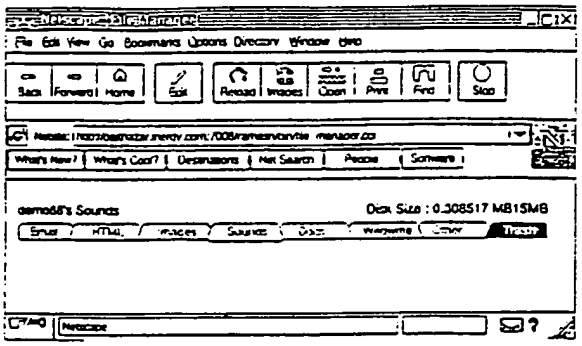


FIG. 13

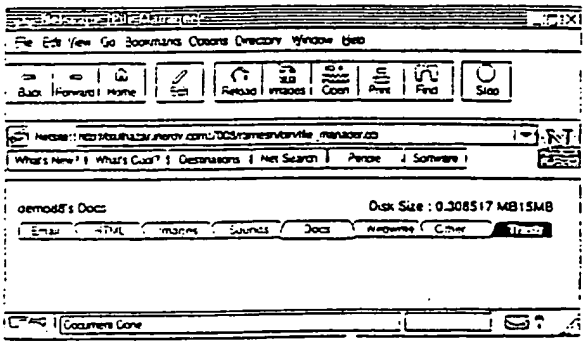


FIG. 14

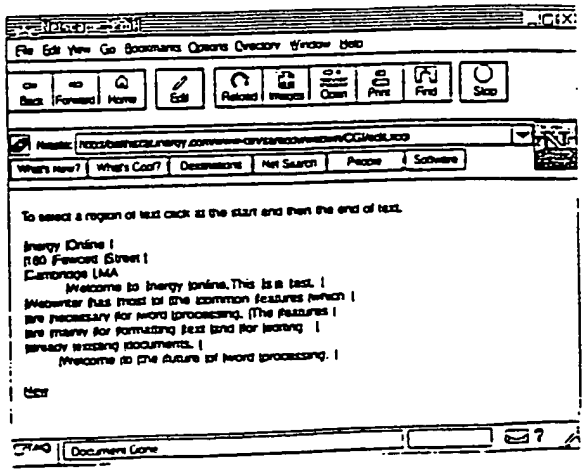


FIG. 15

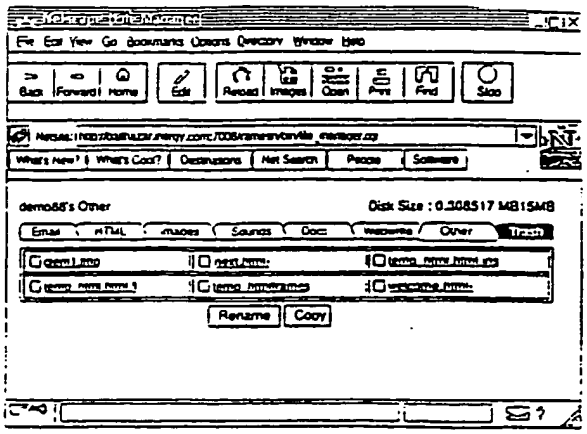


FIG. 16

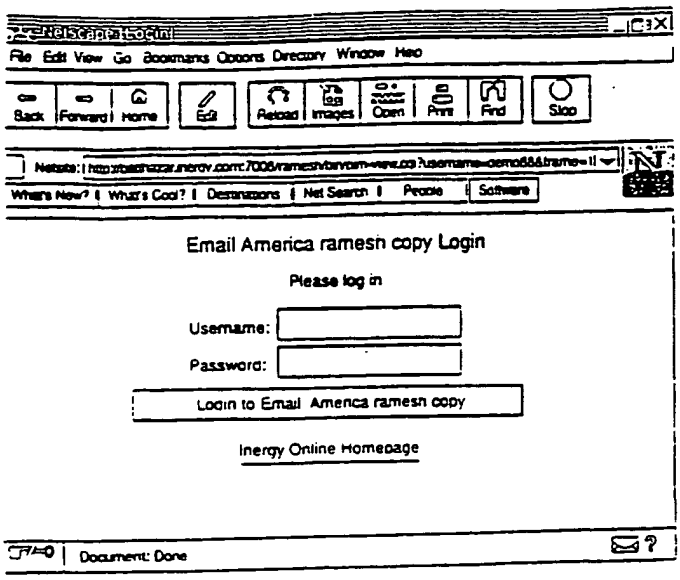


FIG. 17

```
<html>
<!--Copyright (c) 1997 Inergy Online, Inc. All rights reserved. -->
<!-- $Header: /proj/email-2/lib/common.cgi.xpl,v 1.87 1997/08/18 20:0
<head>
<SCRIPT>
<!--
function userfocus () {
self.document.loginform. username.focus()
}
// -->
</SCRIPT>
<title>login</title>
</head>
<body onLoad="userfocus()" bgcolor="#d5d1c0">
<center>
<h1>Email America ramesh copy Login</h1>
Please log in.
<form action="http://balthazar.inergy.com:7008/ramesh/bin/login.cgi">
Username: <input name=username><br>
Password: <input type=password name=password><br>
<input type=submit name="submit" value="Login to Email America rames
</form>
<a href=http://www.inergy.com target=_top onMouseOver="window.status:
</center>
</body>
</html>
```

FIG. 18

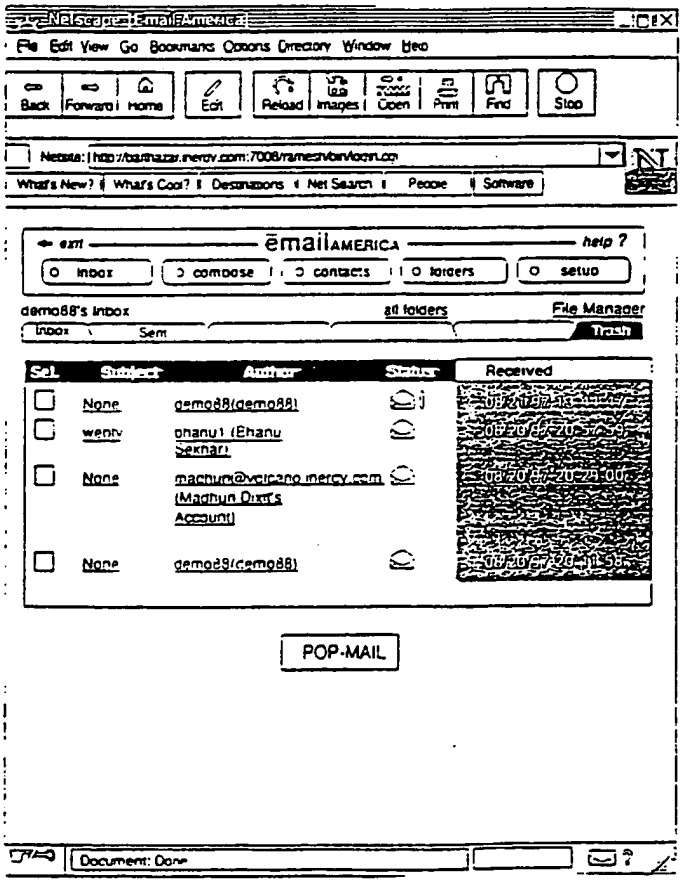


FIG. 19

```
<html>
<!--Copyright (c) 1997 Inergy Online, Inc.  All rights reserved. --
<!-- $Header: /proj/email-2/CGI/mail.xcgi,v 1.36 1997/08/12 21:37:34
<HEAD><TITLE>Email America</TITLE></HEAD>
<body bgcolor="//d5d1c0">
<CENTER>
<BGCOLOR=d5d1c0">
<TABLE CELLSPACING=0 CELLPADDING=0 BORDER=0>
<TR><TD COLSPAN=7><img src="/ramesh/images/toppar.gif" WIDTH=542 HEI
<TR><TD ROWSPAN=2><img src="/ramesh/images/leftbar.gif" WIDTH=3 HEIG
<TD COLSPAN=5><A HREF="/ramesh/bin/logout.cgi" target="_top"><IMG SR
<TD ROWSPAN=2><IMG SRC="/ramesh/images/rightbar.gif" WIDTH=6 HEIGHT=
<TD><A HREF="/ramesh/bin/mailbottom.cgi?username=demo88&frame=1&t=87
<TD><A HREF="/ramesh/bin/compose_screen.cgi?username=demo88&frame=1&
<TD><A HREF="/ramesh/bin/contacts.cgi?username=demo88&frame=1" ><IMG
<TD><A HREF="/ramesh/bin/setup.cgi?username=demo88&frame=1" ><IMG SR
<TD COLSPAN=7><IMG SRC="/ramesh/images/bottombar.gif" WIDTH=542 HEIG

<center><form action="/ramesh/bin/folder_handler.cgi" method=post EN
<INPUT TYPE=HIDDEN NAME="fldr_num" value="">
<INPUT TYPE=HIDDEN NAME="crnt_folder" value="">
 <center><TABLE CELLPADDING=0 CELLSPACING=0 WIDTH=546 BORDER=0><TR><
<input type=hidden name=username_value="demo88">
<input type=hidden name=frame value="1">
</form>
<html><head><title>demo88's mail in folder Inbox</title></head><body
<center><form name=mail_list action="/ramesh/bin/ut1.cgi" method=pos
<input type=hidden name=frame value="1">
<input type=hidden name=sort value="Arrival">
<input type=hidden name=date_format value="%c">
<input type=hidden name=folder value="1000">
<table border=1 cellpadding=0 cellspacing=0 width=546><TR><TD align=
<tr><td valign=top width=25 align=middle><input name="selection" typ
<tr><td valign=top width=25 align=middle><input name="selection" typ
<tr><td valign=top width=25 align=middle><input name="selection" typ
</table></form></td></tr></table>
<center><form name=pop_mail action="/ramesh/bin/pop_mail.cgi" method
<input type=hidden name=frame value="1">
<input type=submit name=pop value="POP-MAIL"></center></form>
</body>
```

FIG. 20

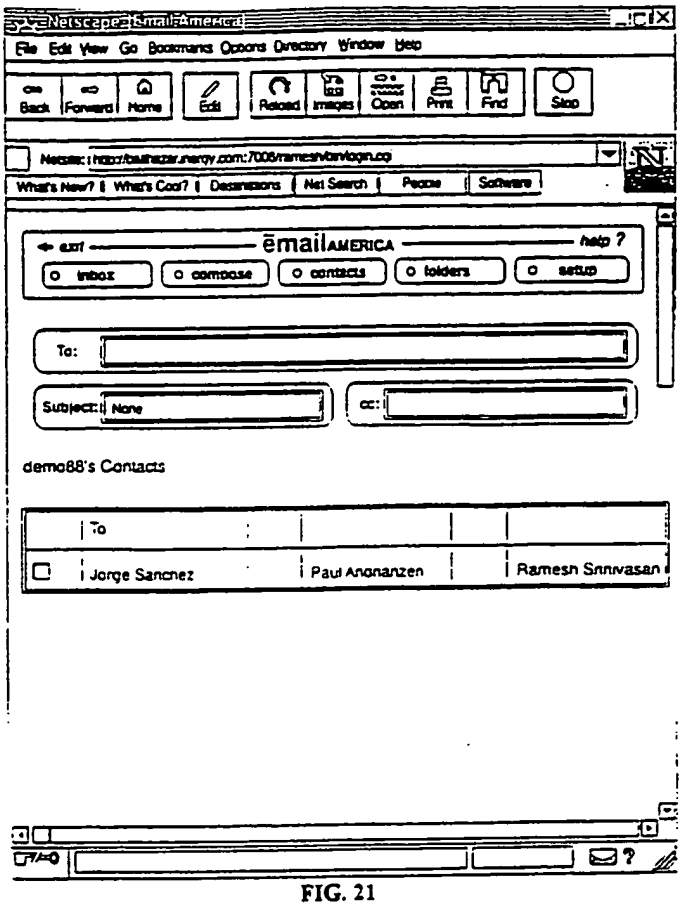


FIG. 21

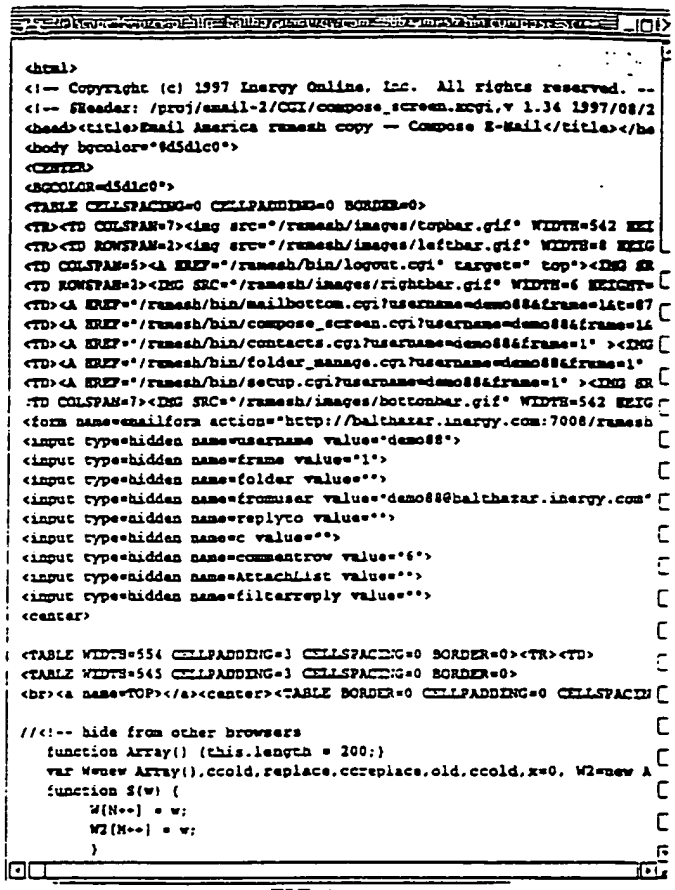


FIG. 22

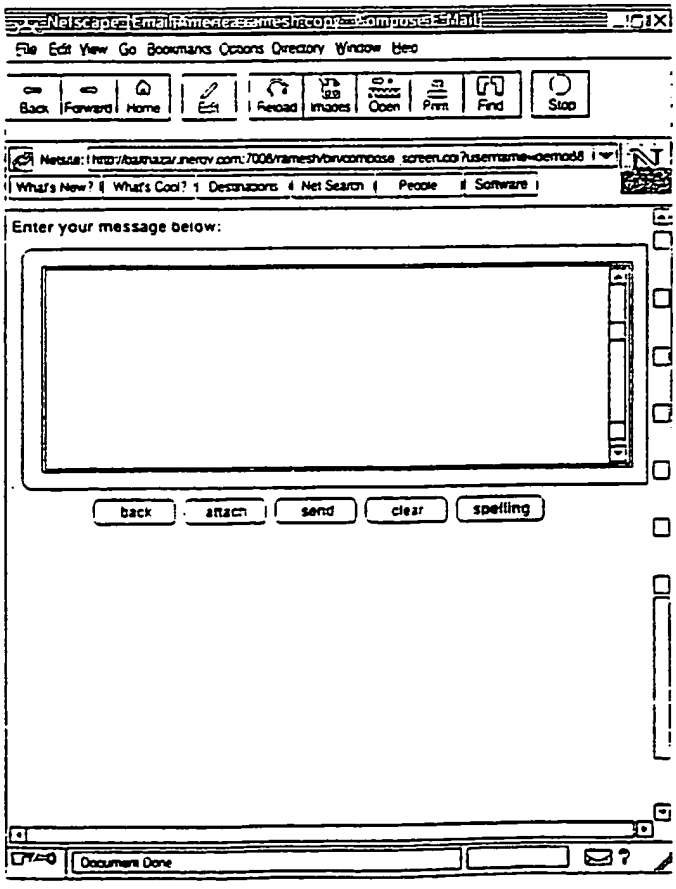


FIG. 23

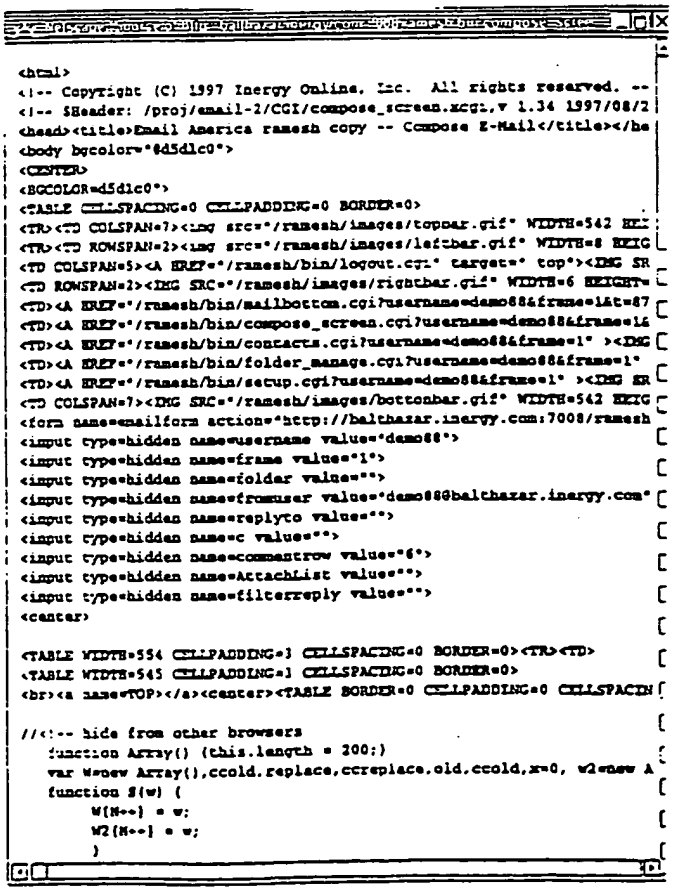


FIG. 24

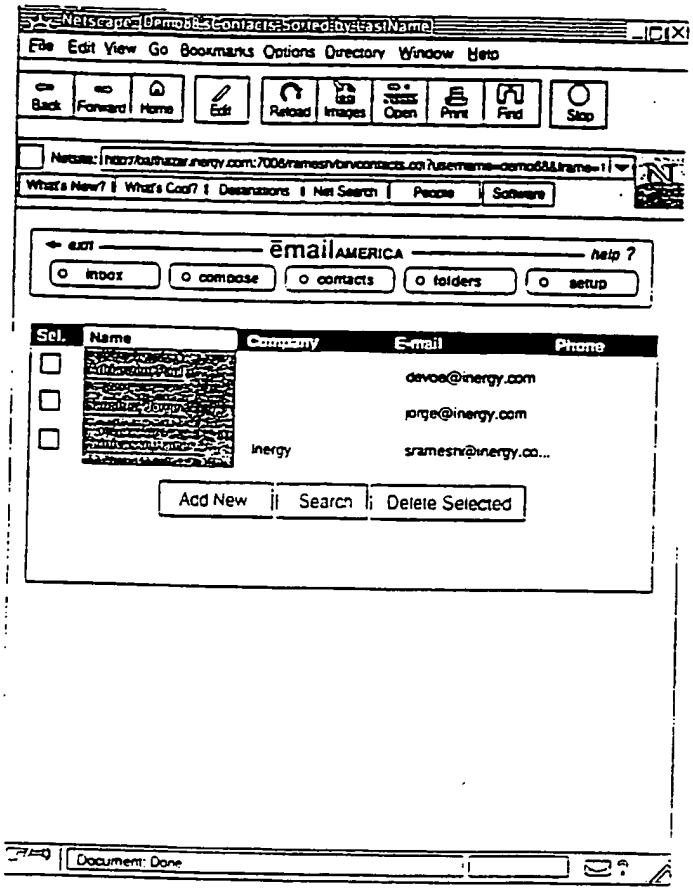


FIG. 25

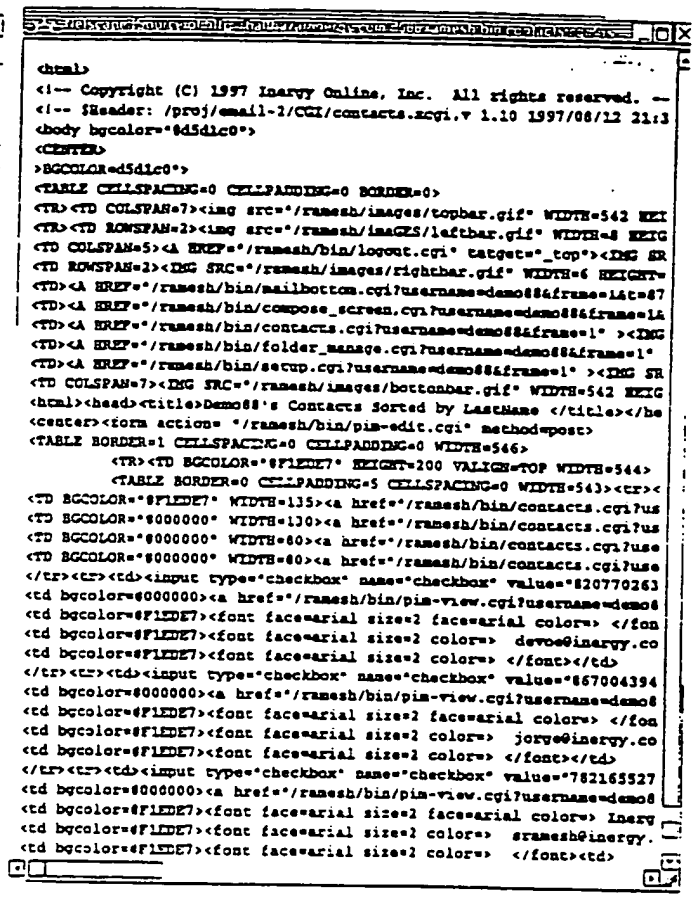


FIG. 26

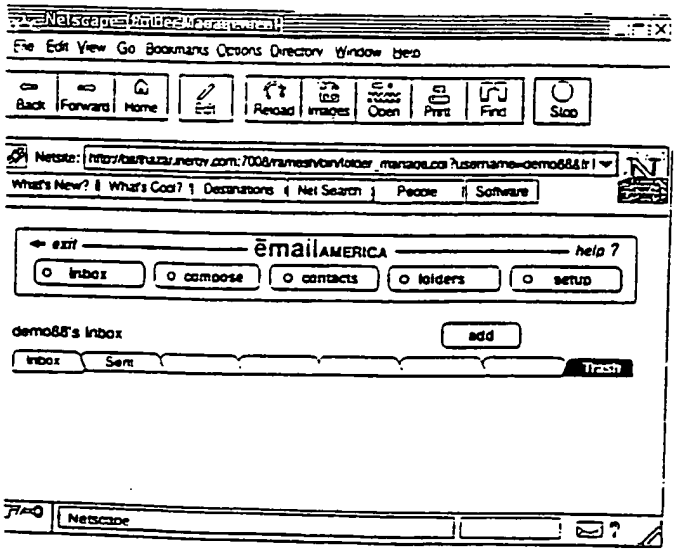


FIG. 27

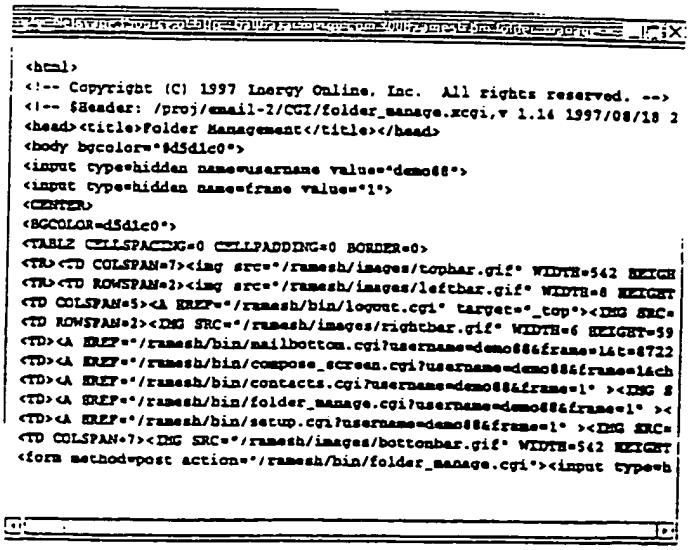


FIG. 28

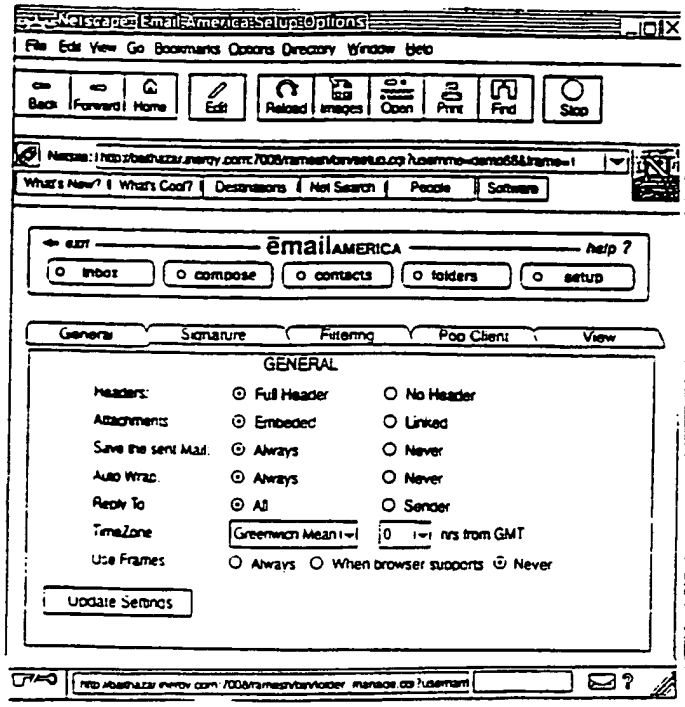


FIG. 29

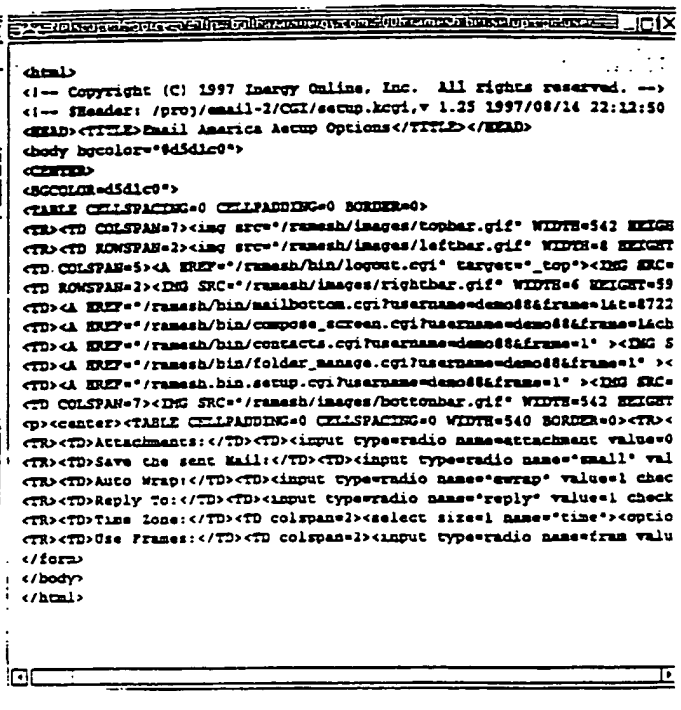


FIG. 30

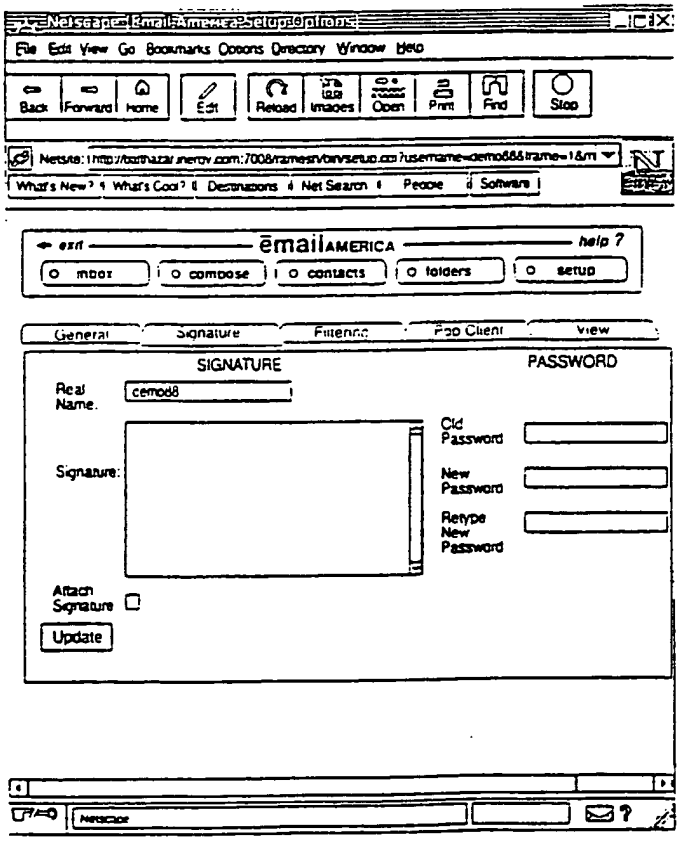


FIG. 31

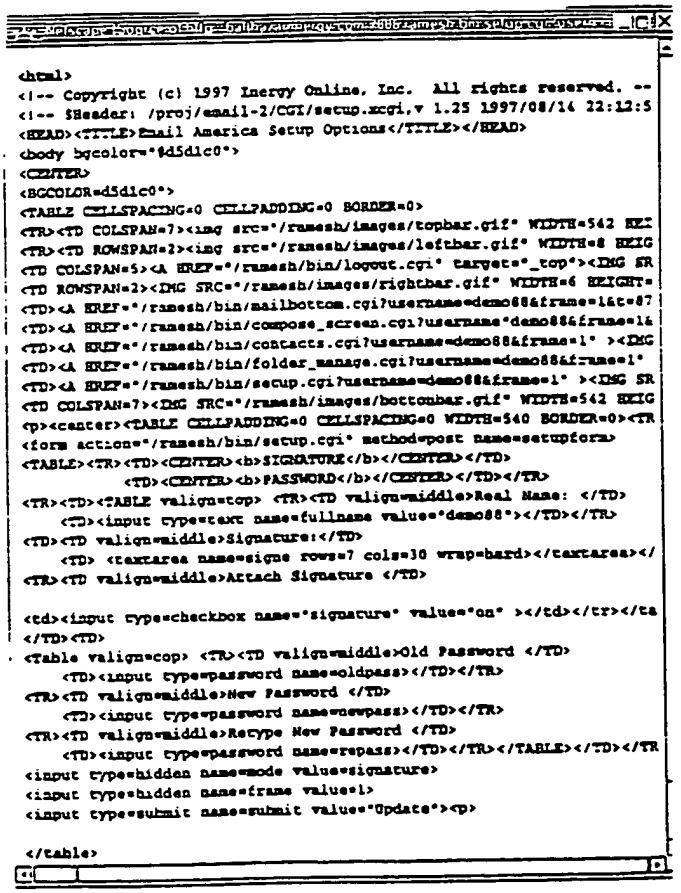


FIG. 32

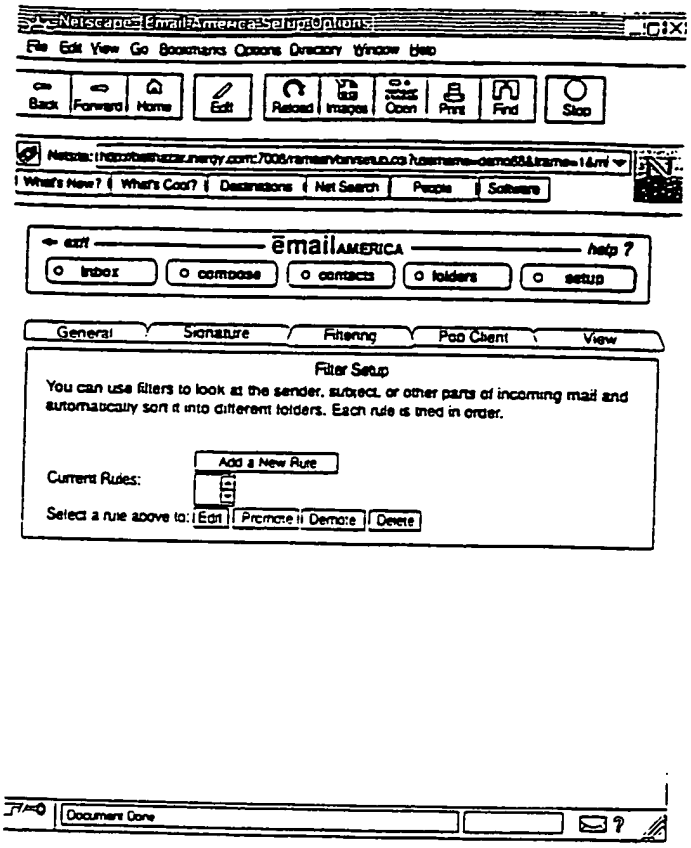
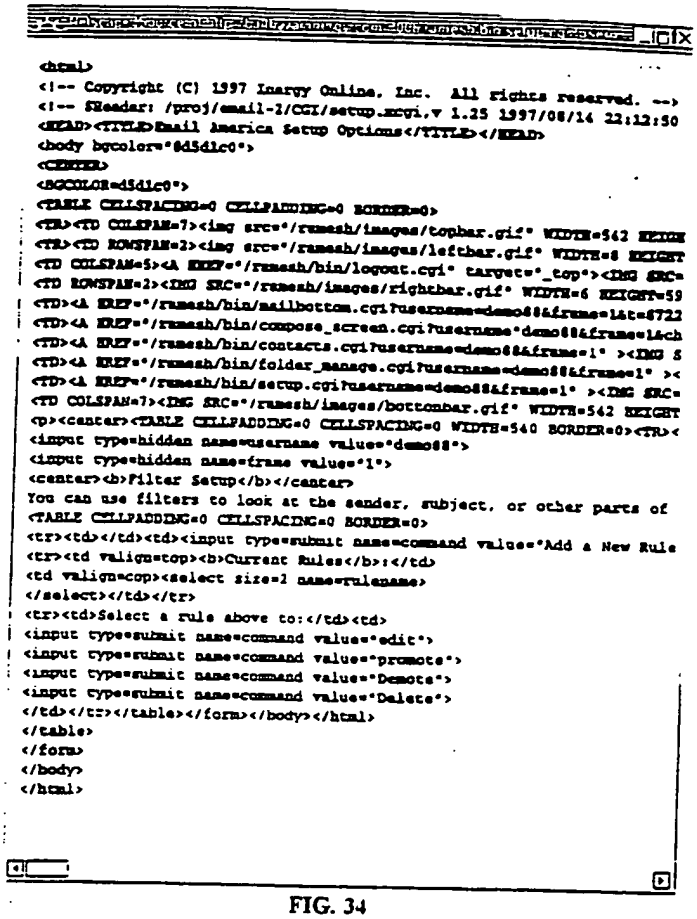


FIG. 33



```
<html>
<!-- Copyright (C) 1997 Inergy Online, Inc.  All rights reserved. -->
<!-- $Header: /proj/email-2/CGI/setup.xcgi,v 1.25 1997/08/14 22:12:50
<HEAD><TITLE>Email America Setup Options</TITLE></HEAD>
<body bgcolor="#d5d1c0">
<CENTER>
<BGCOLOR=#d5d1c0">
<TABLE CELLSPACING=0 CELLPADDING=0 BORDER=0>
<TR><TD COLSPAN=7><img src="/ramesh/images/topbar.gif" WIDTH=542 HEIGH
<TR><TD ROWSPAN=2><img src="/ramesh/images/leftbar.gif" WIDTH=8 HEIGHT
<TD COLSPAN=5><A HREF="/ramesh/bin/logout.cgi" target="_top"><IMG SRC=
<TD ROWSPAN=2><IMG SRC="/ramesh/images/rightbar.gif" WIDTH=6 HEIGHT=59
<TD><A HREF="/ramesh/bin/mailbottom.cgi?username=demo88&frame=1&t=8722
<TD><A HREF="/ramesh/bin/compose_screen.cgi?username=demo88&frame=1&ch
<TD><A HREF="/ramesh/bin/contacts.cgi?username=demo88&frame=1" ><IMG S
<TD><A HREF="/ramesh/bin/folder_manage.cgi?username=demo88&frame=1" ><
<TD><A HREF="/ramesh/bin/setup.cgi?username=demo88&frame=1" ><IMG SRC=
<TD COLSPAN=7><IMG SRC="/ramesh/images/bottonbar.gif" WIDTH=542 HEIGHT
<p><center><TABLE CELLPADDING=0 CELLSPACING=0 WIDTH=540 BORDER=0><TR><
<input type=hidden name=username value="demo88">
<input type=hidden name=frame value="1">
<center><b>Filter Setup</b></center>
You can use filters to look at the sender, subject, or other parts of
<TABLE CELLPADDING=0 CELLSPACING=0 BORDER=0>
<tr><td></td><td><input type=submit name=command value="Add a New Rule
<tr><td valign=top><b>Current Rules</b>:</td>
<td valign=top><select size=2 name=rulename>
</select></td></tr>
<tr><td>Select a rule above to:</td><td>
<input type=submit name=command value="edit">
<input type=submit name=command value="promote">
<input type=submit name=command value="Demote">
<input type=submit name=command value="Delete">
</td></tr></table></form></body></html>
</table>
</form>
</body>
</html>
```

FIG. 34

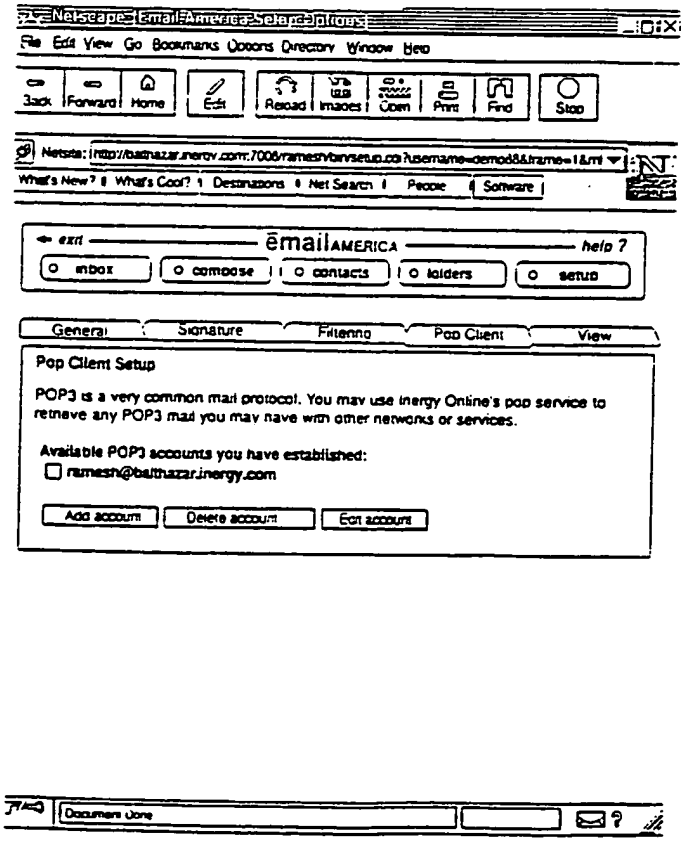
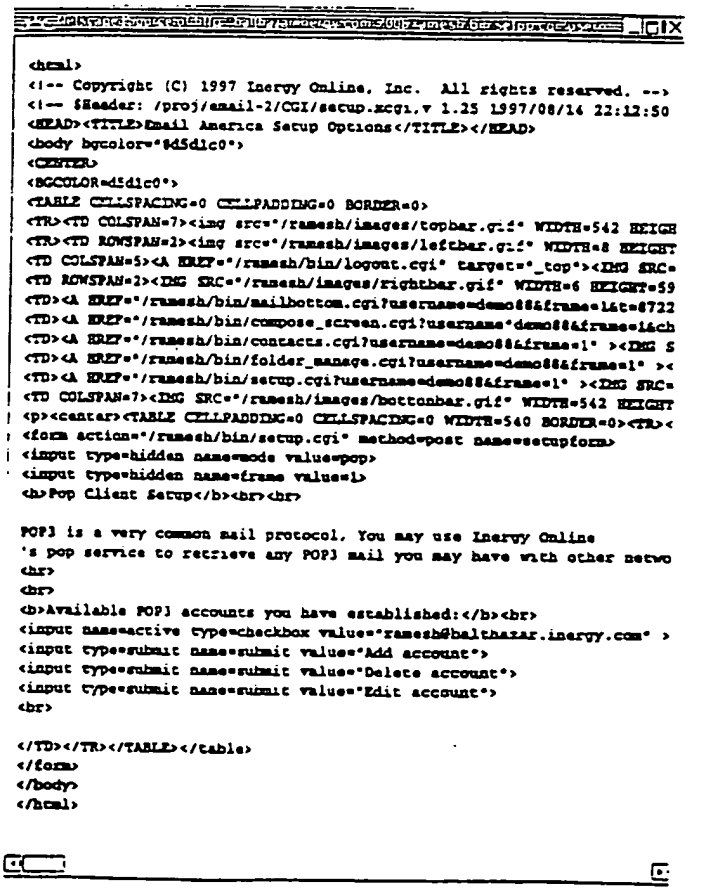


FIG. 35



```
<html>
<!-- Copyright (C) 1997 Inergy Online, Inc.  All rights reserved. -->
<!-- $Header: /proj/email-2/CGI/setup.xcgi,v 1.25 1997/08/14 22:12:50
<HEAD><TITLE>Email America Setup Options</TITLE></HEAD>
<body bgcolor="#d5d1c0">
<CENTER>
<BGCOLOR=#d5d1c0">
<TABLE CELLSPACING=0 CELLPADDING=0 BORDER=0>
<TR><TD COLSPAN=7><img src="/ramesh/images/topbar.gif" WIDTH=542 HEIGH
<TR><TD ROWSPAN=2><img src="/ramesh/images/leftbar.gif" WIDTH=8 HEIGHT
<TD COLSPAN=5><A HREF="/ramesh/bin/logout.cgi" target="_top"><IMG SRC=
<TD ROWSPAN=2><IMG SRC="/ramesh/images/rightbar.gif" WIDTH=6 HEIGHT=59
<TD><A HREF="/ramesh/bin/mailbottom.cgi?username=demo88&frame=1&t=8722
<TD><A HREF="/ramesh/bin/compose_screen.cgi?username=demo88&frame=1&ch
<TD><A HREF="/ramesh/bin/contacts.cgi?username=demo88&frame=1" ><IMG S
<TD><A HREF="/ramesh/bin/folder_manage.cgi?username=demo88&frame=1" ><
<TD><A HREF="/ramesh/bin/setup.cgi?username=demo88&frame=1" ><IMG SRC=
<TD COLSPAN=7><IMG SRC="/ramesh/images/bottonbar.gif" WIDTH=542 HEIGHT
<p><center><TABLE CELLPADDING=0 CELLSPACING=0 WIDTH=540 BORDER=0><TR><
<form action="/ramesh/bin/setup.cgi" method=post name=setupform>
<input type=hidden name=mode value=pop>
<input type=hidden name=frame value=1>
<b>Pop Client Setup</b><br><br>

POP3 is a very common mail protocol. You may use Inergy Online
's pop service to retrieve any POP3 mail you may have with other netwo
<br>
<br>
<b>Available POP3 accounts you have established:</b><br>
<input name=active type=checkbox value="ramesh@balthazar.inergy.com" >
<input type=submit name=submit value="Add account">
<input type=submit name=submit value="Delete account">
<input type=submit name=submit value="Edit account">
<br>

</TD></TR></TABLE></table>
</form>
</body>
</html>
```

FIG. 36

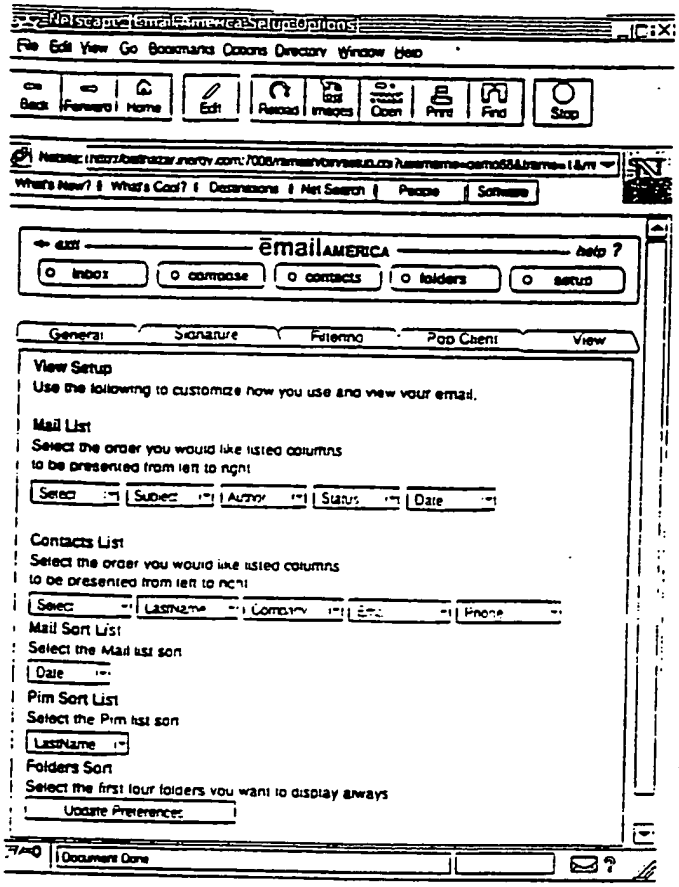


FIG. 37

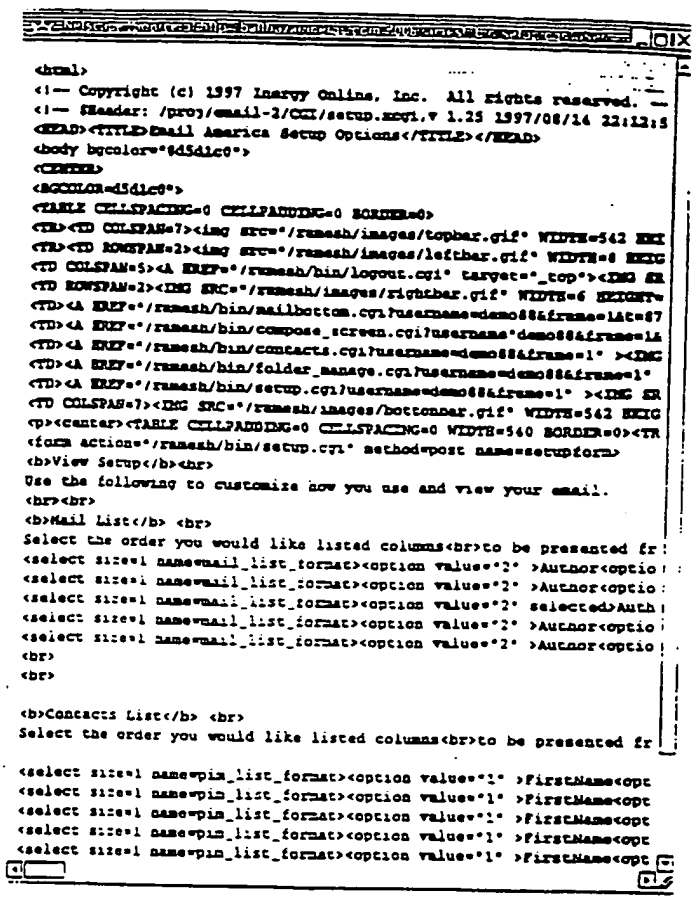


FIG. 38

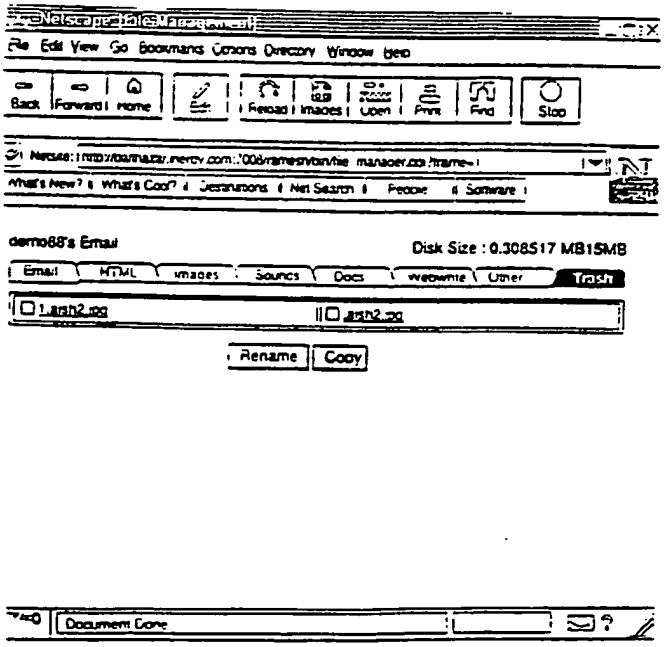


FIG. 39

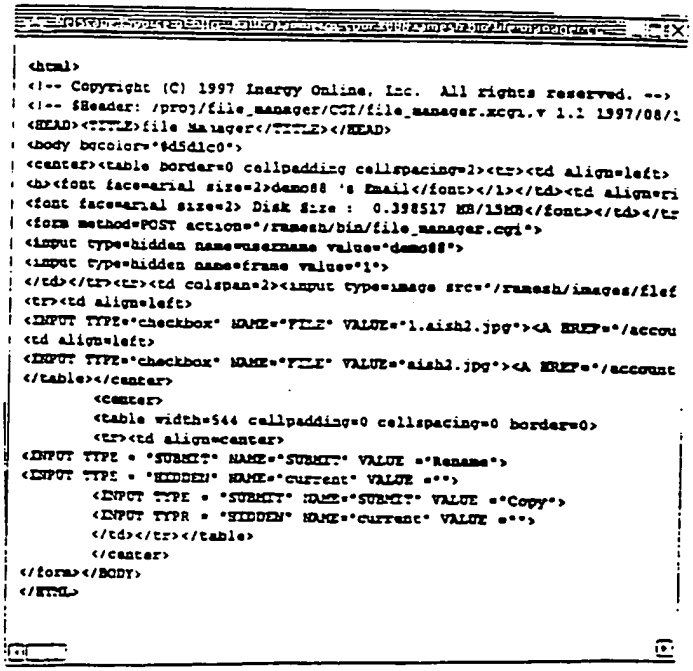


FIG. 40

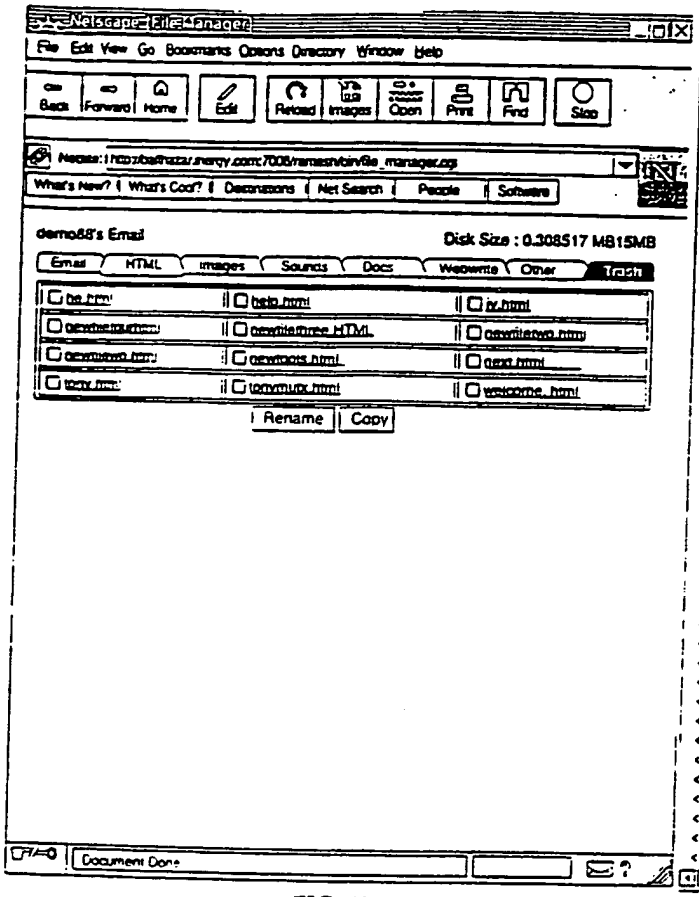


FIG. 41

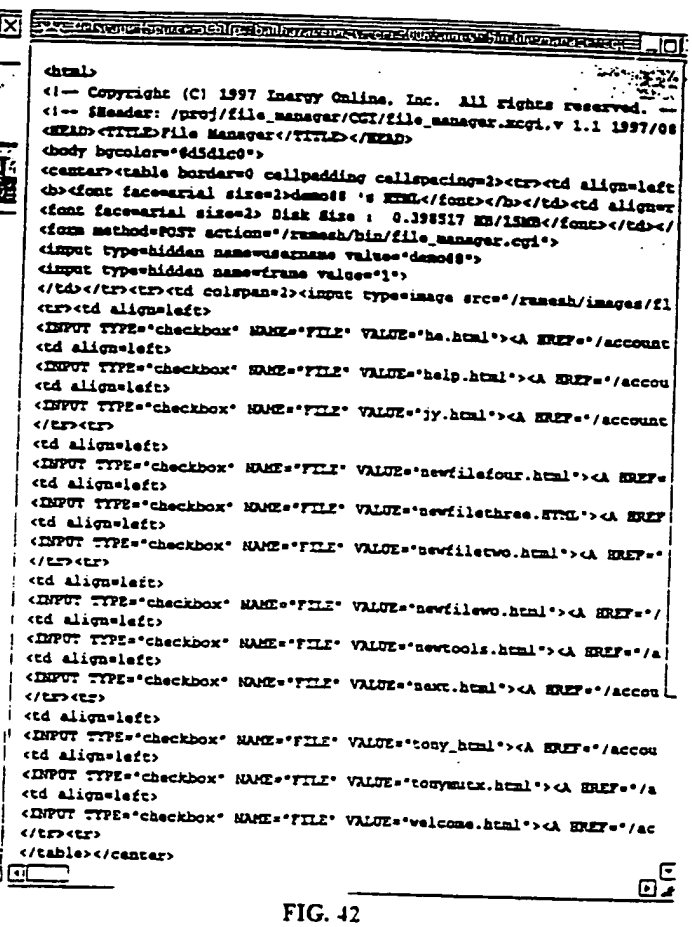


FIG. 42

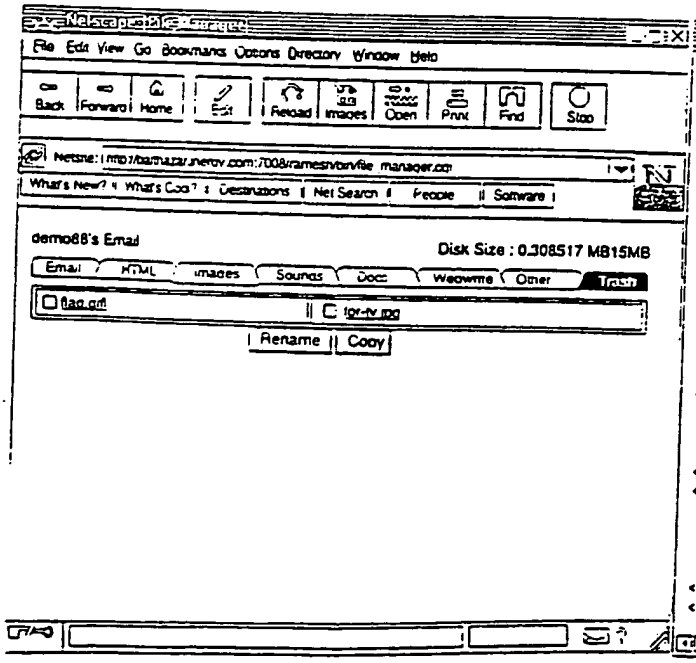


FIG. 43

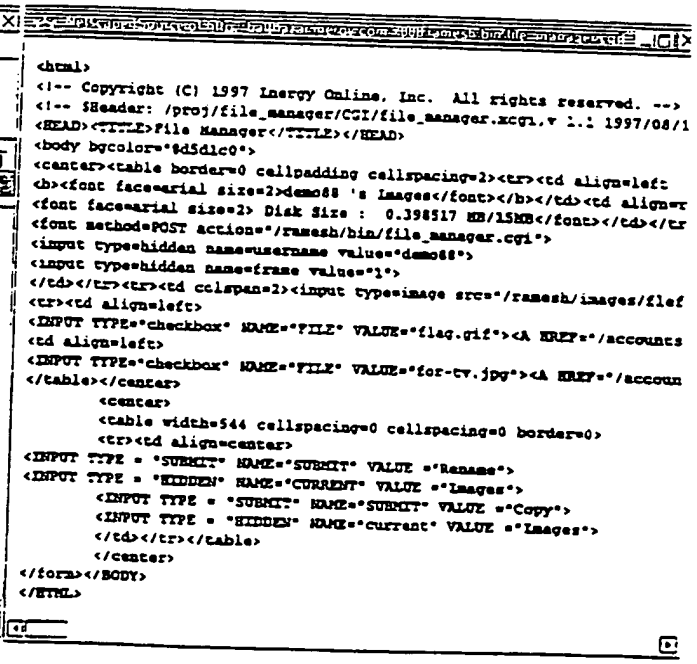


FIG. 44

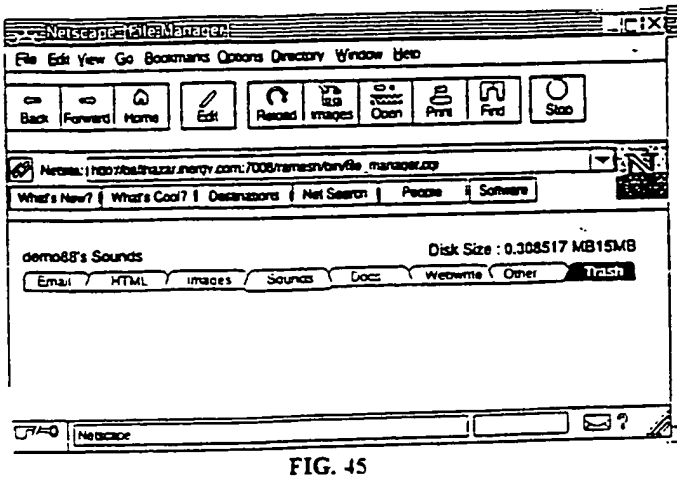


FIG. 45

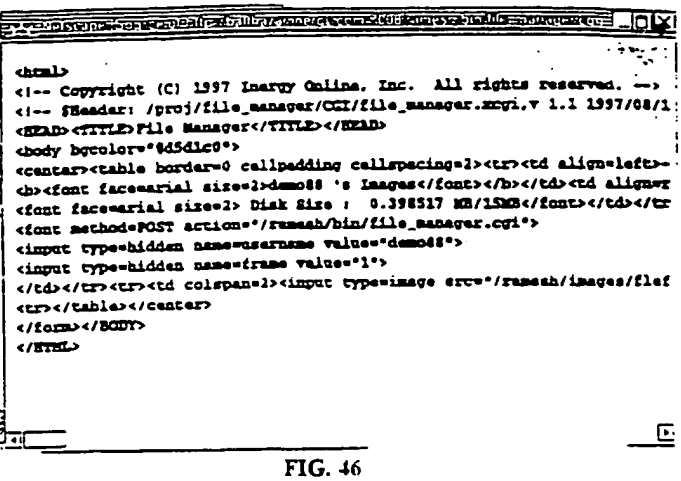


FIG. 46

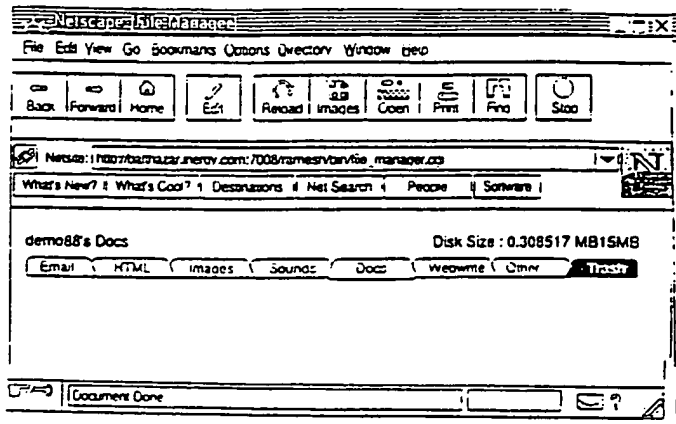


FIG. 47

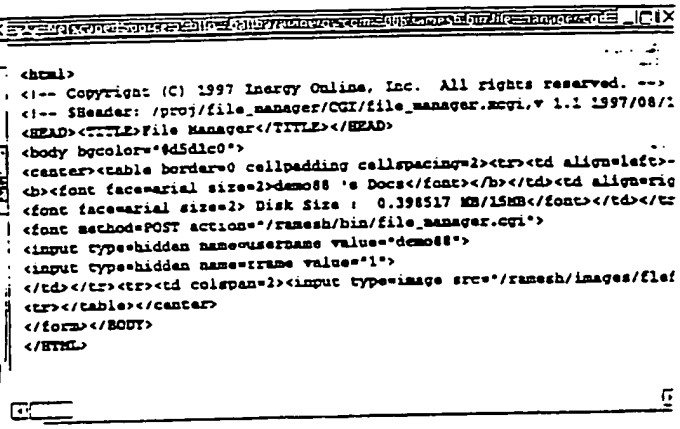


FIG. 48

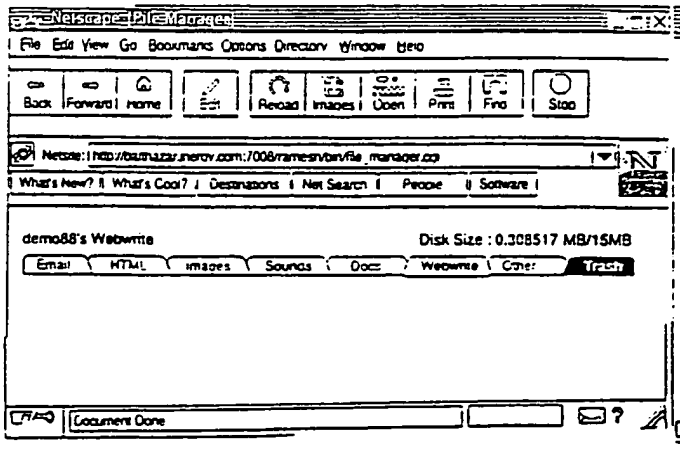


FIG. 49

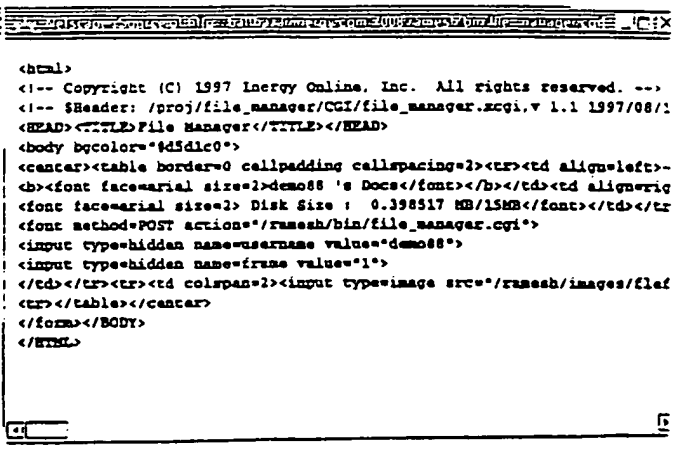


FIG. 50

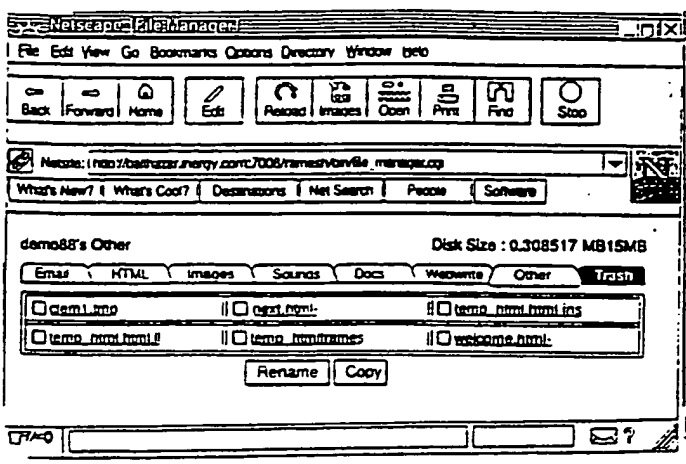


FIG. 51

```
<html>
<!-- Copyright (C) 1997 Inergy Online, Inc.  All rights reserved. -->
<!-- $Header: /proj/file_manager/CGI/file_manager.xcgi,v 1.1 1997/08/1
<HEAD><TITLE>File Manager</TITLE></HEAD>
<body bgcolor="#d5d1c0">
<center><table border=0 cellpadding cellspacing=2><tr><td align=left>
<b><font face=arial size=2>demo88 's Webwrite</font></b></td><td align
<font face=arial size=2> Disk Size :  0.308517 MB/15MB</font></td></tr
<font method=POST action="/rumesh/bin/file_manager.cgi">
<input type=hidden name=username value="demo88">
<input type=hidden name=frame value="1">
</td></tr><tr><td colspan=2><input type=image src="/rumesh/images/file
<tr></table></center>
</form></BODY>
</HTML>
```

FIG. 52

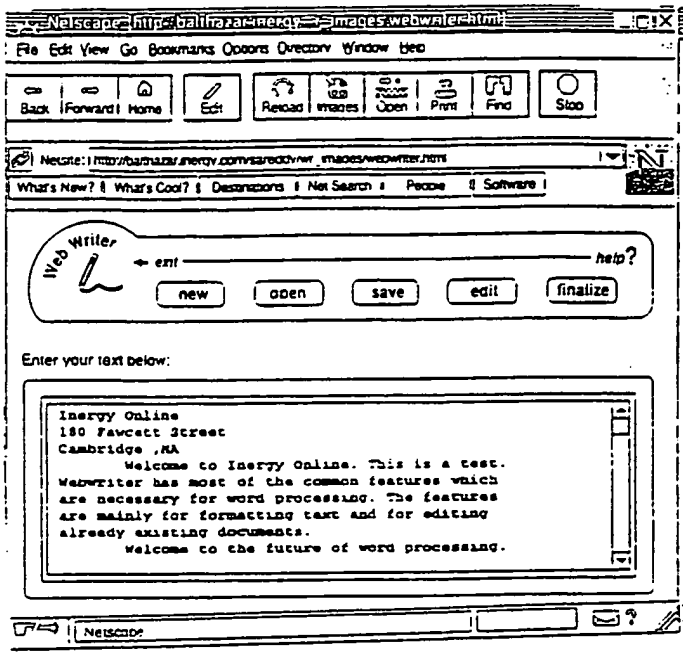


FIG. 53

```
<html>
<head><form method=post  action=/www-bin/sareddy/webwrt/CGI/edit.xcgi>
</head>
<body bgcolor="#d5d1c0"><CENTER>
<TABLE WIDTH=544 BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TABLE WIDTH=540 BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR><TD><IMG SRC="toolbar_logo.gif" WIDTH=69 HEIGHT=79 BORDER=0 ALT="W
<A HREF="http://www.inergy.com/devoe"><IMG SRC="exit_wr_bar.gif "WIDTH
<IMG SRC="left_buttons_wr.gif" WIDTH=22 HEIGHT=19 BORDER=0 ALT=""><A H
<IMG SRC="bottom_bar_wr.gif" WIDTH=449 HEIGHT=9 BORDER=0 ALT=""></TD><
</TABLE>
<TABLE BORDER=0 CELLPADDING=0 CELLSPACING=0>
<TR><TD HEIGHT=50 COLSPAN=2><BR><FONT SIZE=3 FACE=ARIAL>Enter your tex
<TR><TD VALIGN=TOP BGCOLOR=BLACK><IMG SRC="text_border_l2.gif" BORDER=
Inergy Online
180 Fawcett Street
Cambridge ,MA
      Welcome to Inergy Online.This is a test.
Webwriter has most of the common features which
are necessary for word processing. The features
are mainly for formatting text and for editing
already existing documents.
      Welcome to the future of word processing.
</textarea><BR><IMG SRC="top_part.gif" WIDTH=2 HEIGHT=12 BORDER=0 ALT=
<TR><TD VALIGN=BOTTOM BGCOLOR=BLACK><IMG SRC= "text_border_b_l.gif" AL
<TR><TD>./TD></TR>
</form></TABLE><BR></CENTER>
</body>
</html>
```

FIG. 54

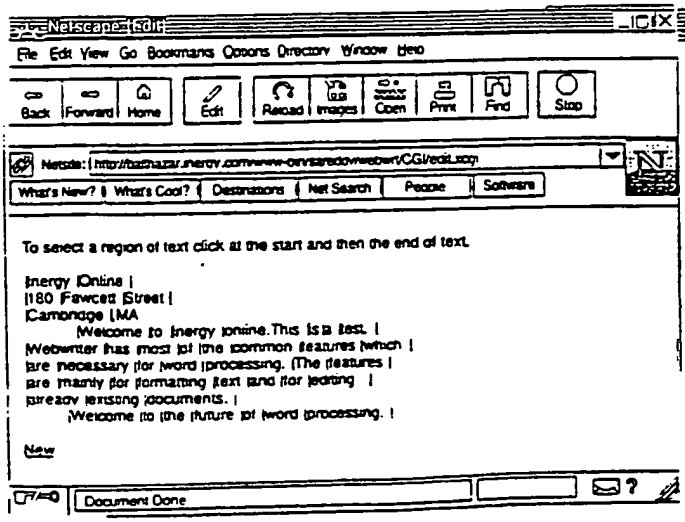


FIG. 55

```
<html>
<HEAD>
<TITLE>Edit</TITLE>
</HEAD>
<BODY>
<BR>
To select a region of text click at the start and then the end of tex
<TR><TD></TD><TD ROWSPAN=50>
<A HREF=format.xcgi?start=0>/</A>Inergy<IMG src=graph.xcgi><A HREF=fo
</BODY>
</HTML>
```

FIG. 56

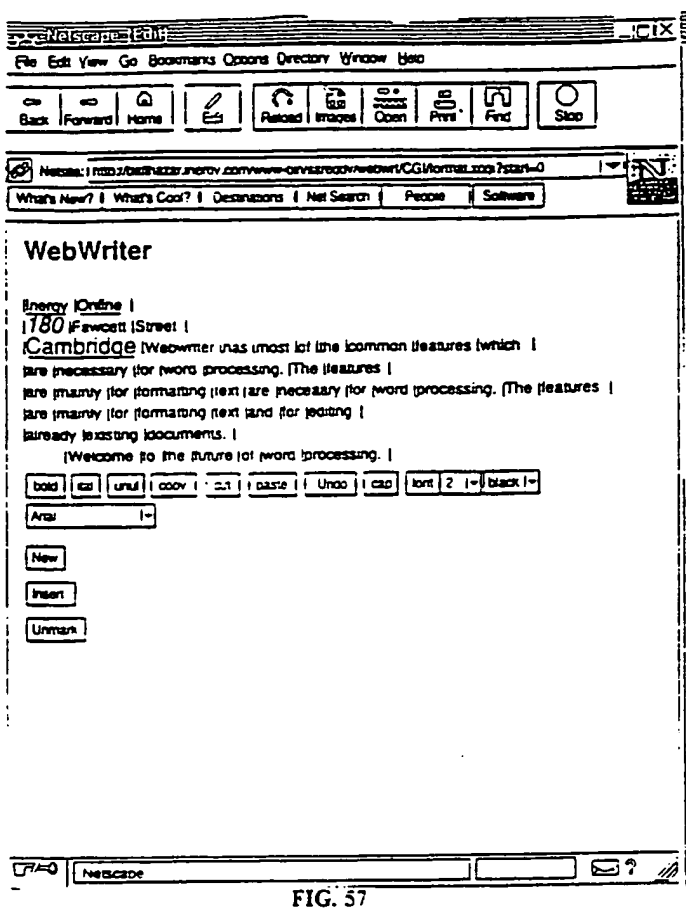


FIG. 57

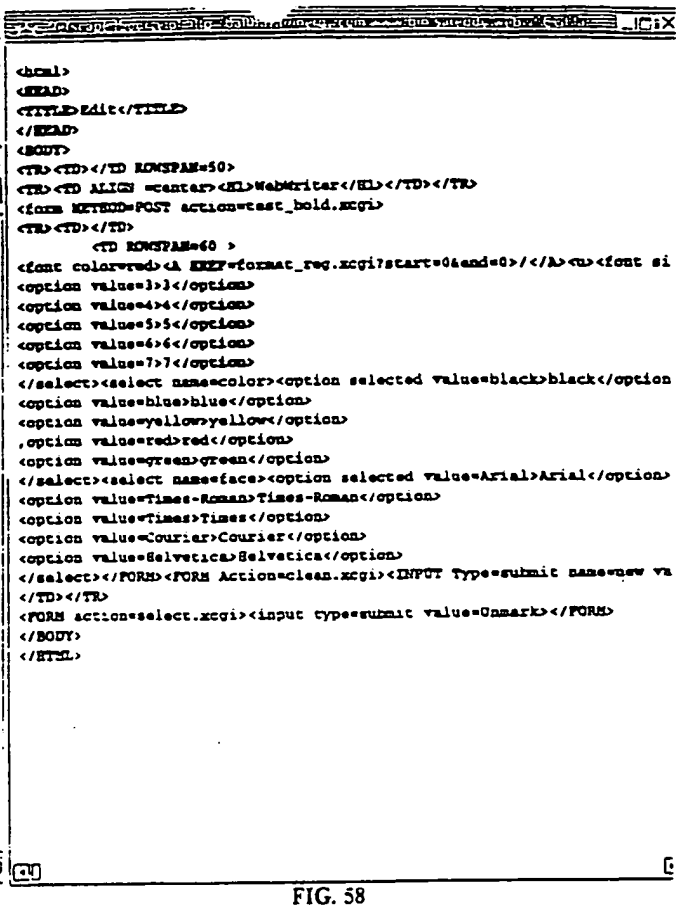


FIG. 58

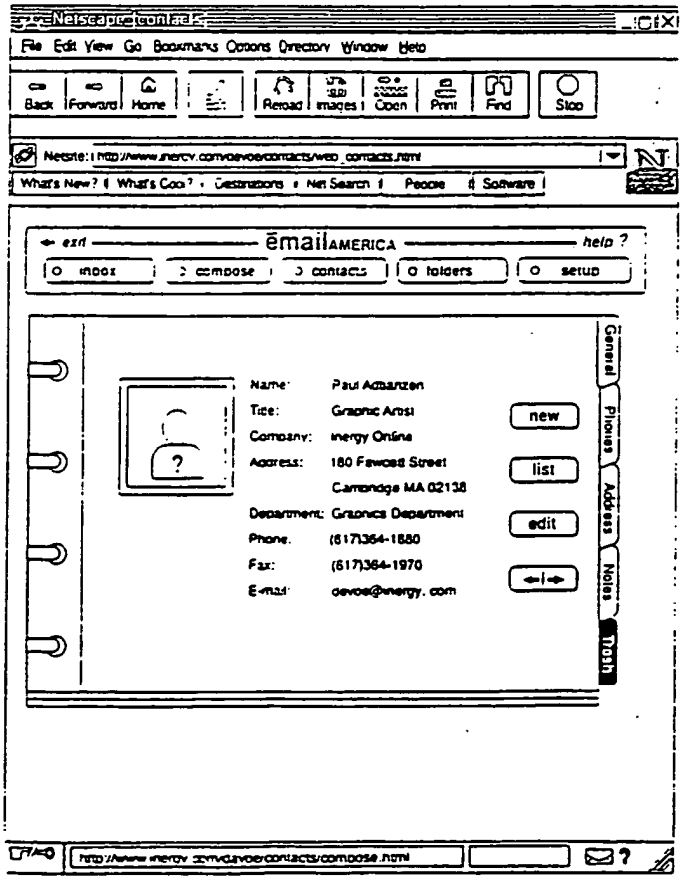


FIG. 59

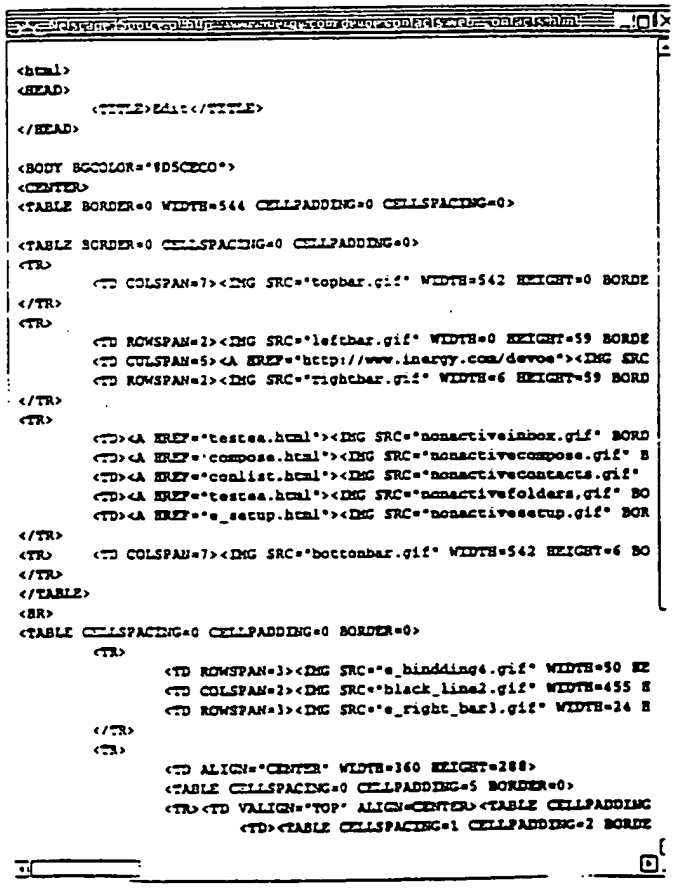


FIG. 60

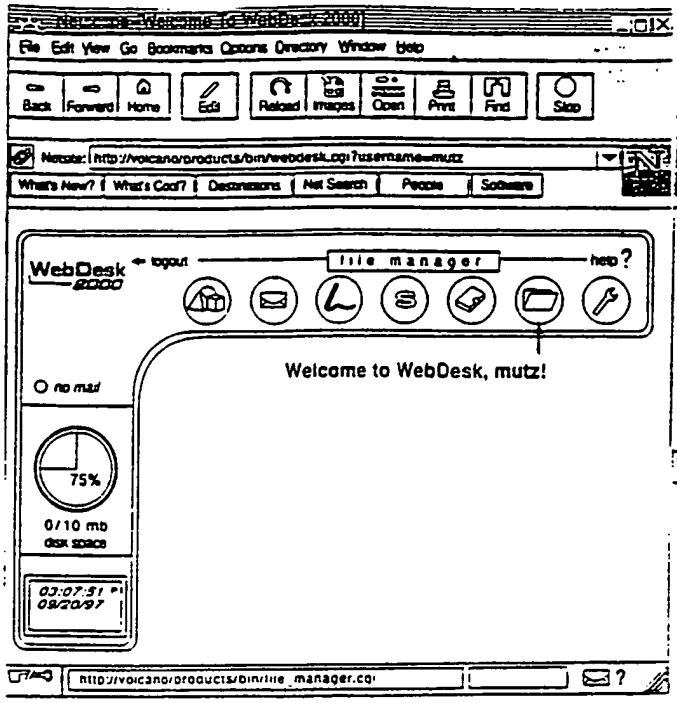


FIG. 61

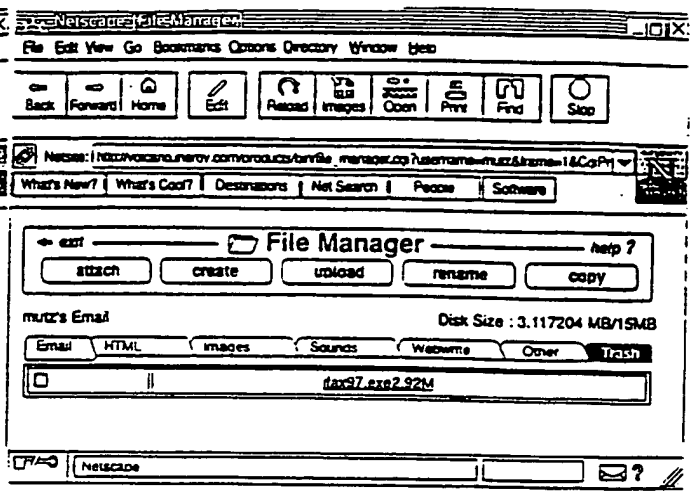


FIG. 62

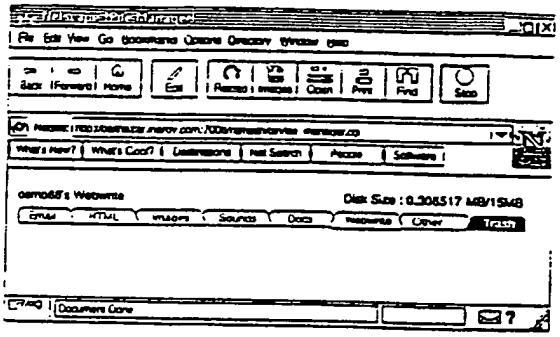


FIG. 63

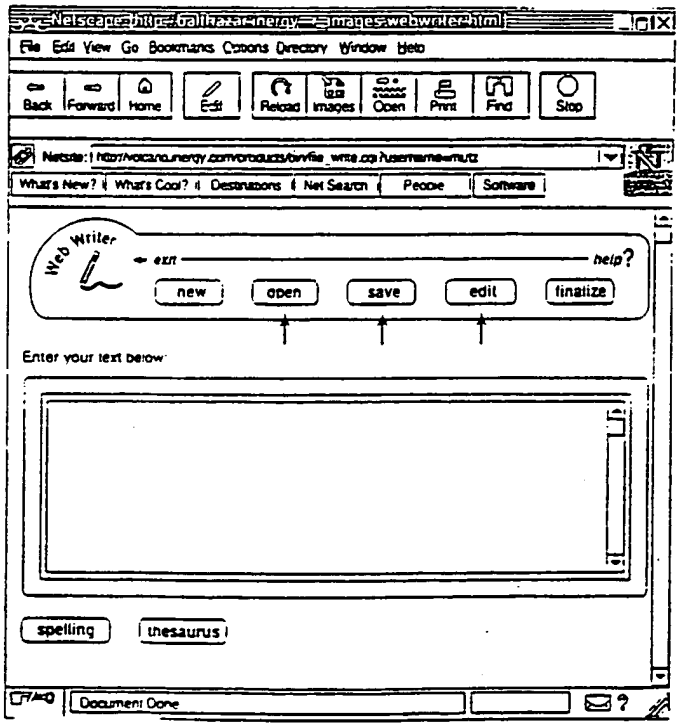


FIG. 64

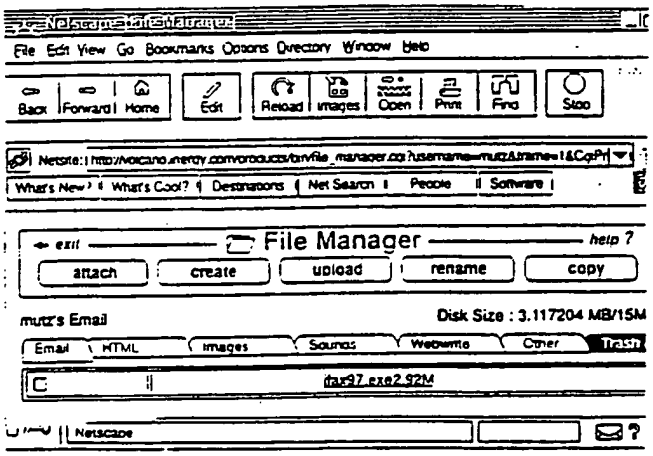


FIG. 65

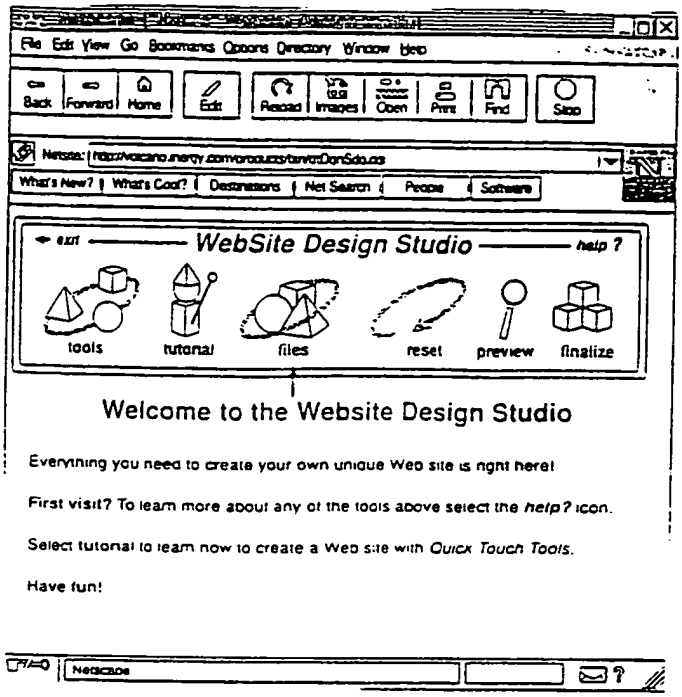


FIG. 66

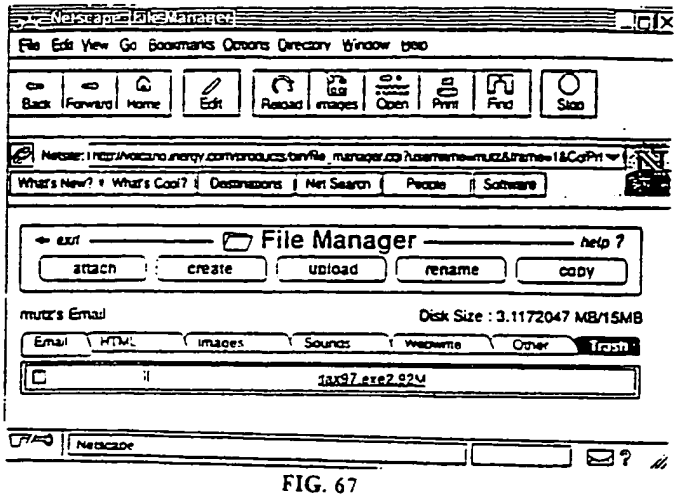


FIG. 67

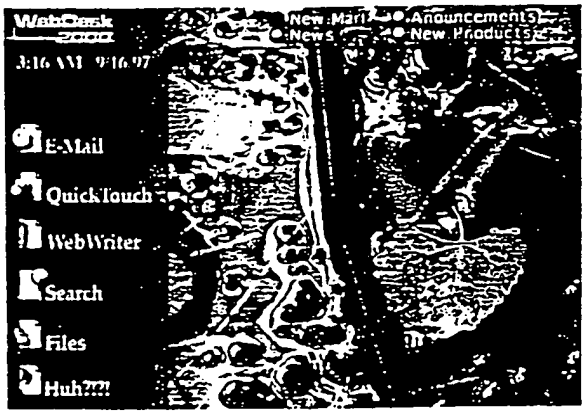


FIG. 69

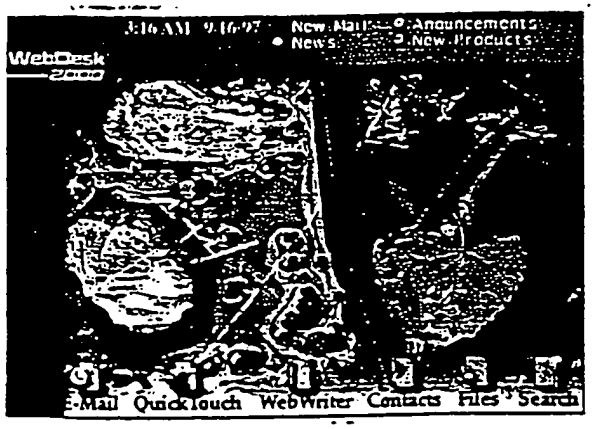


FIG. 71

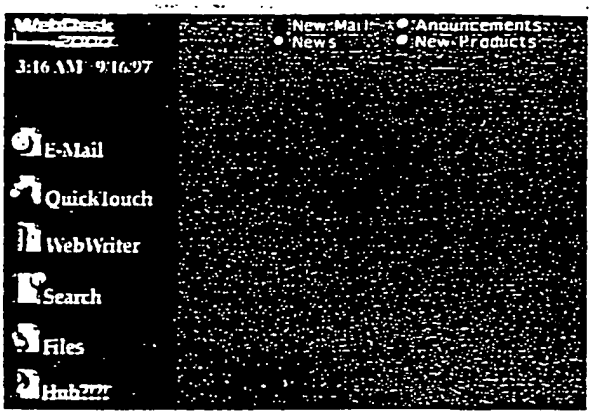


FIG. 68

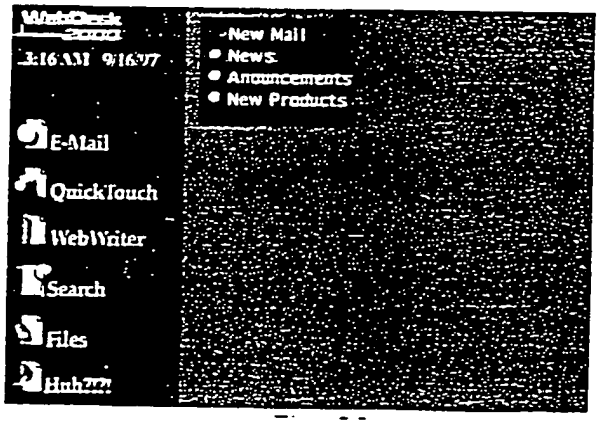


FIG. 70

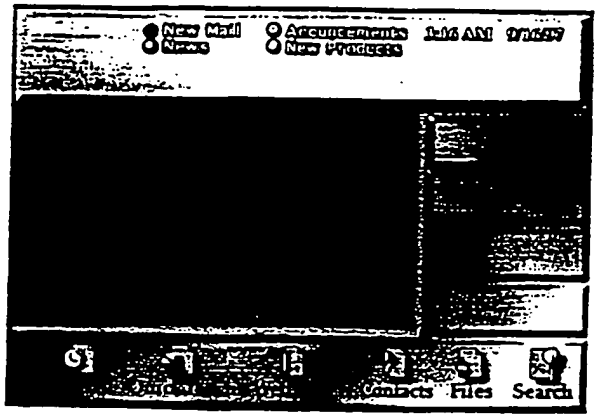


FIG. 72

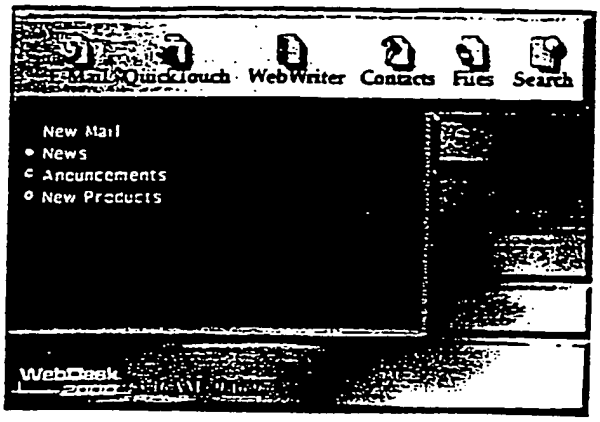


FIG. 73

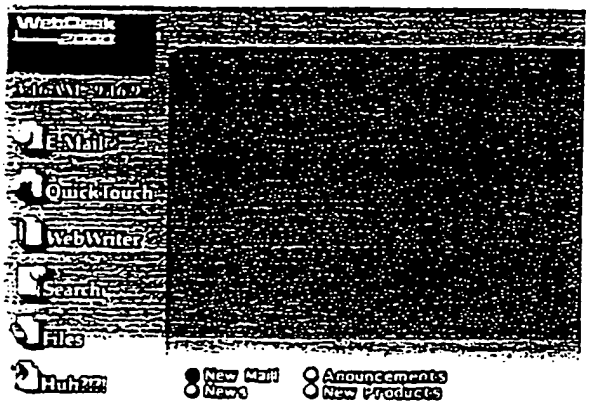


FIG. 74

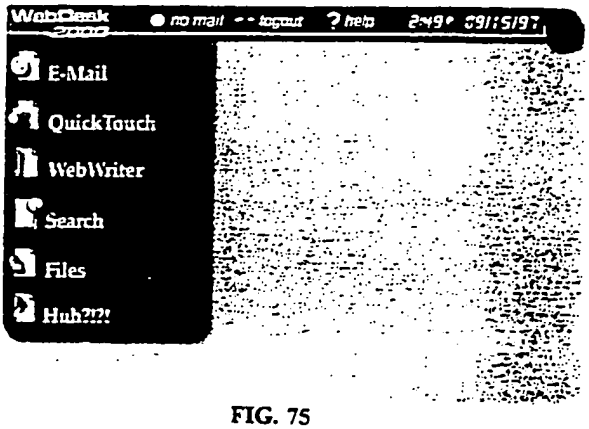


FIG. 75

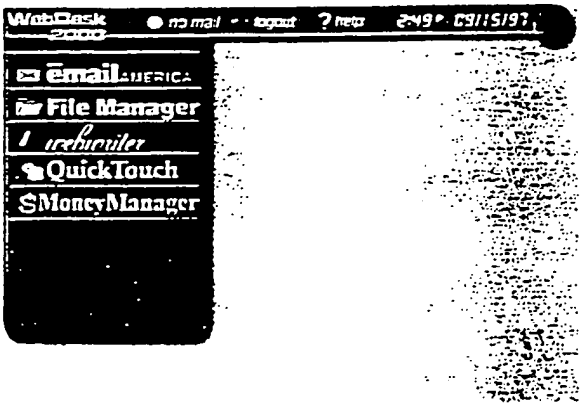


FIG. 76

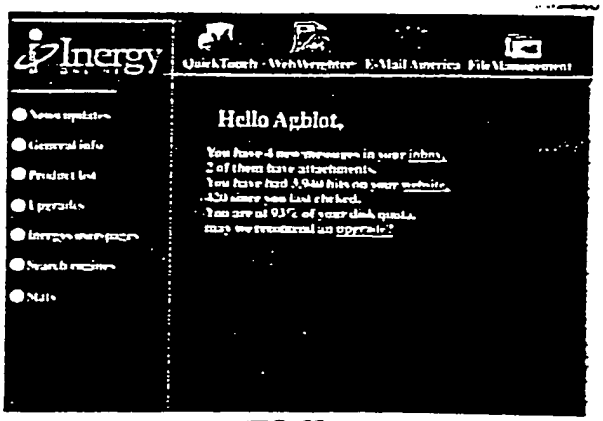


FIG. 77

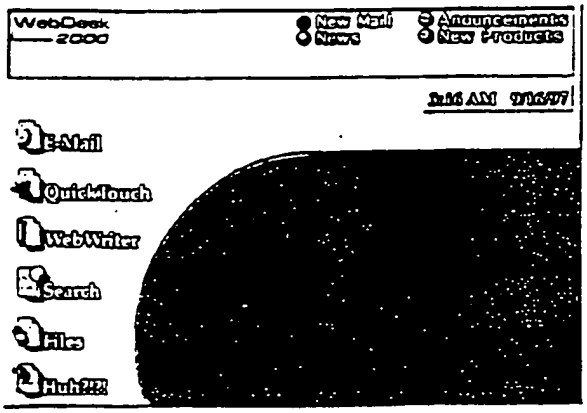


FIG. 78

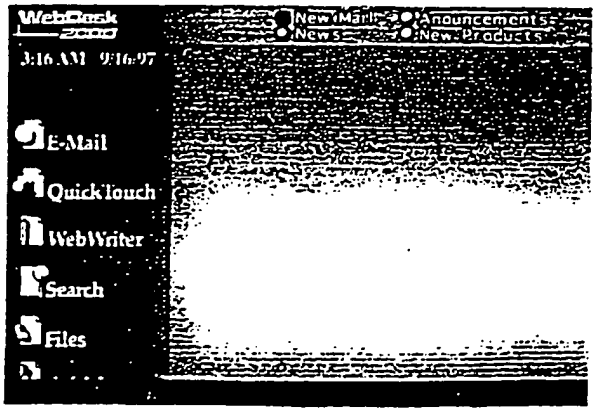


FIG. 79

```
# Copyright (C) 1997 Inergy Online Corp.  A   ights reserved.

use AddINC qw(%%PERL_LIB%%);

use useraccounts qw(QueryUserAccounts);

use File::Copy 'cp';

require 'folder-lib.pl';
require 'viewer.pl';
require 'quota.pl';
require 'upload.pl';

use DbCommonfields ;
use DbMail ;
use CGI;

use vars qw($REALNAME $USERNAME $h_dir $file_count $abs_dir $cgiprog
$print_attach $hostport $frame $current $prev  $MAIL_DB $services) ;


sub file_manager_setup {
    $SUBMIT = $Query->param('SUBMIT');
    $destfile = $Query->param('destfile');
    $SUBMIT1 = $Query->param('SUBMIT1');
    $prev = $Query->param('prev');
    $calling_prog = $Query->param('calling_prog');
    @select_file = $Query->param('FILE');
    @newfile = $Query->param('NEWFILE');
    @trfile = $Query->param('TRASH_FILE');
    @oldfile = $Query->param('OLDFILE');
    $cgiprog = $Query->param('CgiProg');
    $notinhere = $Query->param('notinhere');
    if (!defined($cgiprog)) {
      $cgiprog = $ENV{HTTP_REFERER};
    }
    $print_attach = $Query->param('print_attach');
    $userdir = homedir($USERNAME);
    $h_dir = homedir($USERNAME);
    $his_folder = "$h_dir/www";
}

sub file_manager{
&file_manager_setup;
@mn =
("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"
)[(localtime)[4]];
@dy = (00..30)[(localtime)[3]];
@yer = (localtime)[5];
@day = $dy." ".$mn." ".$yer ." EST";
my ($day) = `date`;
$day =~ s/\d\d:\d\d:\d\d.*//;
$size =  check_dir($h_dir);
$services = QueryUserAccounts($Query, $username, 'service');
my ($display_quota, $mb_converter);
$mb_converter = 1000000;
if ($services eq "") {
    $display_quota = 10;
}if($services =~ /S/i){
    $display_quota = 2 ;

}if($services =~ /M/i){
    $display_quota = 25;
}if($services =~ /P/i || $services =~ /E/i){
    $display_quota = 10;
}if($services =~ /B/i){
    $display_quota = 50;
}
$size = $size / $mb_converter;
$size = int $size;

if ($services eq "") {
    $services = "EIWIM";
}



my($current) = @_;
my(@all);
@all = ('Email', 'HTML', 'Images', 'Sounds',  'Webwriter', 'Other',
'Trash');
$fm_attach = $Query->param('fm_Attach.x');
$fm_rename = $Query->param('fm_Rename.x');
$fm_copy = $Query->param('fm_Copy.x');
$fm_cancel_button = $Query->param('fm_cancel_button.x');
$fm_upload = $Query->param('fm_Upload.x');
$fm_upload_button = $Query->param('fm_upload_button.x');
$fm_create_button = $Query->param('fm_create_button.x');
$fm_continue_button = $Query->param('fm_continue_button.x');
$fm_edit = $Query->param('fm_Edit.x');
$fm_create = $Query->param('fm_Create.x');
$go = $Query->param('GO.x');
$dir_path = $Query->param('dir_path');
$params = $userdir."/.fileparams";
    %folhash = ('HTML'   => ".html, .htm",
                'Images' => ".gif, .jpg, .jpeg",
               'Sounds' => ".mid, .wav, .au, .gsm, .ra",
               'Webwriter' => ".wbt" );
if($go){
  $SUBMIT1 = "DONE";
  }
if ($cgiprog =~ /compose/i || $cgiprog =~ /inbox/i || $cgiprog =~
/WebDesk/i){
$functimage = "fm_attach.jpg";
$functname = "fm_Attach";
$strcr = "<img  SRC=\"/%%MAIL_IMAGEDIR%%/fm_create.jpg\" WIDTH=98
HEIGHT=35 BORDER=0 ALT=\"create\" HSPACE=0>";
}
else{
$functimage = "fm_edit.jpg";
$functname = "fm_Edit";
$strcr = "<input type=image  SRC=\"/%%MAIL_IMAGEDIR%%/fm_create.jpg\"
WIDTH=98 HEIGHT=35 BORDER=0 ALT=\"create\" HSPACE=0 name=fm_Create
nocursor>";
}

if ($fm_attach){
    $SUBMIT = 'Attach';
  }
elsif($fm_rename){
    $SUBMIT = 'Rename';
}
elsif ($fm_upload){
    $SUBMIT = 'Upload';
}
elsif ($fm_create){
    $SUBMIT = 'Create';
}
elsif ($fm_copy){
    $SUBMIT = 'Copy';
}
elsif ($fm_edit){
    $SUBMIT = 'Edit';
}
# if this routine is called for the first time $cgiprog will have a
value from the query
  # If so, get the rest of the needed vars and store them to a temp
file.
  if ($cgiprog && !defined($notinhere) && !defined($dir_path))  {
    $notinhere = 1;
    @tos = $Query->param('tos');
    @ccs = $Query->param('ccs');
    @bccs = $Query->param('bccs');
    $subject = $Query->param('subject');
    $subject=defined($subject)?$subject:"None";
    $AttachList = $Query->param('AttachList') ;
    $Priority = $Query->param('Priority') ;
    $folder = $Query->param('folder');
    $command = $Query->param('c');
    $comments = $Query->param('comments');
    $replyto = $Query->param('replyto');
    $print_attach = $Query->param('print_attach');
    $actarg = $Query->param('filterreply');



    open OUTPUT, ">$params" || die "could not open $params \n";

    if( @tos ) {
      foreach (@tos) {
        $ToAdd = $ToAdd.",".$_ ;
      }
    }
    if( @ccs ) {
      foreach (@ccs) {
        $CcAdd = $CcAdd.",".$_ ;
      }
    }
    if( @bccs ) {
      foreach (@bccs) {
        $BccAdd = $BccAdd.",".$_ ;
      }
    }
    $SaveCompose = join("\c_",
$ToAdd,$CcAdd,$BccAdd,$subject,$Priority,$replyto,$print_attach,$actarg,
$cgiprog,$comments) ;
    print OUTPUT $SaveCompose ;
    close(OUTPUT);
  }

&print_toolbar;

 if ($SUBMIT eq 'Create'){
  $exithere =  &file_create;
    if($exithere){
    goto EXITHERE;
    }
 }

 @print "<br>\n";
 if ($SUBMIT eq 'Upload'){
    print "</form>";
  $exithere =  &file_upload;
    if($exithere){
    goto EXITHERE;
    }
 }
 if ($SUBMIT eq 'Empty'){
  $exithere =  &file_empty;
    if($exithere){
    goto EXITHERE;
    }
 }


 if ($select_file[0] && ($current eq $prev  && !defined($SUBMIT))){
      foreach $sel_file(@select_file){
        $sel_file = '';
      }
 }
 elsif($trfile[0] && ($current eq $prev) && !defined($SUBMIT)){
      foreach $trfile(@trfile){
        $trfile = '';
      }
 }
 if (($select_file[0]) && ($prev eq 'Email') && (($current eq
'HTML')||($current eq 'Images') || ($current eq 'Sounds')||($current eq
'Webwriter') ||($current eq 'Other'))){
      print "<table width=544 border=0 cellpadding=0
cellspacing=0><tr><td><font face=arial size=2>The folder to which each
file has been moved is displayed depending upon their extention. .gif,
.jpg, .jpeg => Images. .htm, .html => HTML. .gsm, .mid, .ra, .au, .wav
=> Sounds. All other files were moved to the Other
folder.</font></td></tr></table>";
      $current = $prev;
      &move_from_email ;
 }
 elsif (($select_file[0]) && ($current eq 'Email') && (($prev eq
'HTML')||($prev eq 'Images') || ($prev eq 'Sounds')||($prev eq
'Webwriter') ||($prev eq 'Other'))){
      $current = $prev;
       &move_from_web;
 }
 elsif (($select_file[0]) && ($current eq 'HTML') && (($prev eq 'Images')
|| ($prev eq 'Sounds')||($prev eq 'Webwriter') ||($prev eq 'Other'))){

      print "<table width=544 cellpadding=0 cellspacing=0
border=0><tr><td><font face=arial size=2>Only $folhash{$current} files
can be moved to the $current folder.</font></td></tr></table>";
      $current = $prev;
      foreach $sel_file(@select_file){
        $sel_file = '';
```

FIG. 80

```
elsif (($select_file[0]) && ($current eq '___ges') && (($prev eq 'HTML')
|| ($prev eq 'Sounds')||($prev eq 'Webwriter') ||($prev eq 'Other'))){
     print "<table width=544 cellpadding=0 cellspacing=0
border=0><tr><td><font face=arial size=2>Only $folhash{$current} files
can be moved to the $current folder.</font></td></tr></table>";
     $current = $prev;
     foreach $sel_file(@select_file){
       $sel_file = '';
     }
}
elsif (($select_file[0]) && ($current eq 'Sounds') && (($prev eq 'HTML')
|| ($prev eq 'Images')||($prev eq 'Webwriter') ||($prev eq 'Other'))){
     print "<table width=544 cellpadding=0 cellspacing=0
border=0><tr><td><font face=arial size=2>Only $folhash{$current} files
can be moved to the $current folder.</font></td></tr></table>";
     $current = $prev;
     foreach $sel_file(@select_file){
       $sel_file = '';
     }
}
elsif (($select_file[0]) && ($current eq 'Webwriter') && (($prev eq
'HTML') || ($prev eq 'Images')||($prev eq 'Sounds') ||($prev eq
'Other'))){
     print "<table width=544 cellpadding=0 cellspacing=0
border=0><tr><td><font face=arial size=2>Only $folhash{$current} files
can be moved to the $current folder.</font></td></tr></table>";
     $current = $prev;
     foreach $sel_file(@select_file){
       $sel_file = '';
     }
}
elsif (($select_file[0]) && ($current eq 'Other') && (($prev eq 'HTML')
|| ($prev eq 'Images')||($prev eq 'Sounds') ||($prev eq 'Webwriter'))){
     print "<table width=544 cellpadding=0 cellspacing=0
border=0><tr><td><font face=arial size=2>Only files  with "." extentions
other then .gif, .jpg, .jpeg, .html, .htm, .wbt, .qsm, .mid, .rs, .au,
.wav will be available in the Other folder.</font></td></tr></table>";
     $current = $prev;
     foreach $sel_file(@select_file){
       $sel_file = '';
     }
}

print Tm "Vars : prev=$prev, $destfile,$SUBMIT, $SUBMIT1,
$cgiprog,$select_file[0]";
close Tm;

if (!$current) {
      $current = 'Email';
}

  print qq|<center><table border=0 cellpadding=0 cellspacing=0
width=544><tr><TD><TD HEIGHT=7> </TD></TR><td align=left valign=top
height=23>\n|;
  print qq|<b><font face=arial size=2> $day </font></b></td><td
align=right valign=top height=23>\n|;
  print qq|<font face=arial size=2> Disk Size:   $size MB/$display_quota

MB</font></td></tr></table>\n|;
  print qq|<table width=544 cellpadding=0 cellspacing=0 border=0>|;
  standard_form();
  if($select_file[0] || $trfile[0]){
  print_tabs($prev, @all);
  file_type($prev);
}
else{
  print_tabs($current, @all);
  file_type($current);
}

$file_count=0;
if (($current eq 'Email') || (($prev eq 'Email') && ($select_file[0]))){
$abs_dir = "$h_dir/mail/attachments";
$web_dir = "%%ATTACHMENT_PATH%%/mail/attachments";
}
else{
$abs_dir = "$h_dir/www";
$his_folder = "$h_dir/www";
$web_dir = "%%ATTACHMENT_PATH%%/www";
}


opendir(DIR, "$abs_dir") || die ("Can't open directory");

$count=1;
print "</td></tr></table><CENTER><TABLE BORDER=1 cellpadding=0
cellspacing=2 width=544>\n";
print "<tr>";
if ($SUBMIT1 eq 'DONE' && $SUBMIT eq 'Rename'){
&funct_rename;
}
elsif ($SUBMIT1 eq 'DONE' && $SUBMIT eq 'Copy'){
&funct_copy;
}
elsif (($current eq 'Trash') && ($select_file[0])){
&funct_move;
}
elsif(($current) && (@trfile) && ($current ne 'Trash')){
&funct_unmove;
}
  if($select_file[0] || $trfile[0]){
  $current = $prev;
}
print "<input type=hidden name=prev value=$current>";
print "<input type=hidden name=notinhere value=notinhere>";
while (@file1=readdir(DIR)){
@file2=sort(@file1);
$cnt=0;
foreach $file(@file2){
# if($file =~ /\s/){
#    $displayfile =~ s/\s/%20/g ;
# }
 if ($current ne 'Trash'){
  $t_check = trash_check($file);
  if($t_check){
    my $fsize = StrimDecimals("/WWW/$web_dir/$file") ;
#   $fsize= int $fsize;
  if($current eq 'Other'){
    if ($file !~
/^[.]{1,}[.*]html$|[.*]htm$|.. |gif$|[.*]jpg$|[.*]jpeg$|[.*]au$|[.*]wav$|[.
*]wbt$|[.*]mid$|[.*]qsm$|[.*]bmp$|temp_html.temp|temp_html.user/i){
       $file_count=~;
     if($select_file[$cnt] eq $file){
       foreach $sel_file(@select_file){
         if($sel_file eq $file){
          if($SUBMIT eq 'Rename'){
            print qq|<td align=middle colspan=2>\n<INPUT TYPE="TEXT"
NAME="NEWTILE" VALUE="$sel_file"><input type=hidden name=OLDFILE
value=$file><INPUT type=hidden name=SUBMIT value=Rename></td>\n|;
          }
          elsif($SUBMIT eq 'Copy'){
           $select_file1 = "cp_of_".$sel_file ;
            print qq|<td align=middle>\n<INPUT TYPE="checkbox"
NAME="FILE" VALUE="$file"></td><td align=left><A
HREF="$web_dir/$file"><font size=2 face=arial>$file
<i>$fsize</i></font></a><br>\n|;
           print qq|<INPUT type=hidden name=SUBMIT value=Copy><INPUT
TYPE="TEXT" NAME="NEWTILE" VALUE="$select_file1" size=12><input
type=hidden name=OLDFILE value=$file></td>\n|;
          }
         }
       }
          $cnt=~;
     }
     else{
      print qq|<td align=middle>\n<INPUT TYPE="checkbox" NAME="FILE"
VALUE="$file"></td><td align=left><A HREF="$web_dir/$file"><font size=2
face=arial> $file <i>$fsize</i></font></a></td>\n|;
     }

     if ($count== 3)
     {
       print ("<\/tr><tr>\n");
     $count =0 ;
     }
     $count = $count + 1;
}
}
 elsif (($file =~
/(.*)$curr1$|(.*)$curr2$|(.*)$curr3$|(.*)$curr4$|(.*)$curr5$/i) &&
($file ne '.') && ($file ne '..')&& ($file ne '.dir_size') &&($file ne
'.config') &&($file ne '.silver') &&($file ne '.platinum') &&($file ne
'.gold') && ($file ne $nocurr1) && ($file ne $nocurr2) && ($file ne
$nocurr3) && ($file ne $nocurr4) && ($file ne $nocurr5)){

    $file_count=~;
      if($select_file[$cnt] eq $file){
        foreach $sel_file(@select_file){
          if($sel_file eq $file){
          if($SUBMIT eq 'Rename'){
            print qq|<td align=middle colspan=2>\n<INPUT TYPE="TEXT"
NAME="NEWTILE" VALUE="$sel_file"><input type=hidden name=OLDFILE
value=$file><input type=hidden name=SUBMIT value=Rename></td>\n|;
          }
          elsif($SUBMIT eq 'Copy'){
           $select_file1 = "cp_of_".$sel_file ;
            print qq|<td align=middle>\n<INPUT TYPE="checkbox"
NAME="FILE" VALUE="$file"></td><td align=left>|;

            if($file =~ /\.wbt$/i){
            print qq|<A
HREF="/%%MAIL_CGI_BIN%%/open_file.cgi?filename=$file">|;
            }
            else{
            print qq|<A HREF="$web_dir/$file">|;
            }
            print qq|<font face=arial size=2>
$file<i>$fsize</i></font></a><br>\n|;
            print qq|\n<input type=hidden name=SUBMIT
value=Copy><INPUT TYPE="TEXT" NAME="NEWTILE" VALUE="$select_file1"
size=12><input type=hidden name=OLDFILE value=$file></td>\n|;
          }
         }
         }

          $cnt=~;
      }
      else{
       #    $file_count=~;
     print qq|<td align=middle>\n<INPUT TYPE="checkbox" NAME="FILE"
VALUE="$file"></td><td align=left>|;
            if($file =~ /\.wbt$/i){
            print qq|<A
HREF="/%%MAIL_CGI_BIN%%/open_file.cgi?filename=$file">|;
            }
            else{
            print qq|<A HREF="$web_dir/$file">|;
            }
    print qq|<font face=arial size=2>$file
<i>$fsize</i></font></a></td>\n|;
      }

      if ($count== 3)
      {
        print ("\<\/tr>\<tr>\n");
        $count = 0;
      }
      $count = $count + 1;
   }
  }
 }
}
closedir(DIR);
if($current eq 'Trash'){
    &print_trash;
  }
print qq|</tr></table></center>\n|;

 &file_process($current);
 print qq|</form>|;
EXITHERE:
}
sub file_type{

my($curr) = @_;

    $nocurr1 = 'saved_temp.html';
    $nocurr2 = 'temp_html.html';
    $nocurr3 = 'bac_up.html';
```

FIG. 81

```
    $nocurr4 = 'preview.html';
    $nocurr5 = 'temp_html.ins';


if ($curr eq 'HTML'){

    $curr1 = 'html';
    $curr2 = 'htm';
    $curr3 = 'HTML';
    $curr4 = 'HTM';
    $curr5 = 'HTM';
}
elsif ($curr eq 'Email'){

    $curr1 = '^[^.]';
    $curr2 = '';
    $curr3 = '';
    $curr4 = '';
    $curr5 = '';
}
elsif ($curr eq 'Sounds'){

    $curr1 = 'gsm';
    $curr2 = 'wav';
    $curr3 = 'mid';
    $curr4 = 'au';
    $curr5 = 'ra';
}
elsif ($curr eq 'Webwriter'){

    $curr1 = 'wbt';
    $curr2 = 'WBT';
    $curr3 = 'WBT';
    $curr4 = 'wbt';
    $curr5 = 'wbt';
}
elsif ($curr eq 'Other'){

    $curr1 = '';
    $curr2 = '';
    $curr3 = '';
    $curr4 = '';
    $curr5 = '';
    $nocurr1 = '.*';
    $nocurr2 = '.dir_size';
    $nocurr3 = '.bkup';
    $nocurr4 = '.*';
    $nocurr5 = '.*';
}
elsif ($curr eq 'Images'){

    $curr1 = 'gif';
    $curr2 = 'jpg';
    $curr3 = 'GIF';
    $curr4 = 'JPG';
    $curr5 = 'jpeg';
}

}
sub file_process{


my($current) = @_;
if ($file_count > 0){
print <<FM1;
        <center>
        <table width=544 cellpadding=0 cellspacing=0 border=0>
        <tr><td align=center>
FM1
  if ($current ne 'Trash'){
   if (($SUBMIT eq 'Rename') || ($SUBMIT eq 'Copy'))
     && (($SUBMIT1 ne 'DONE')) && ($fm_rename || $fm_copy) &&
$select_file[0]
     ){
print <<FM3;
        <INPUT TYPE="IMAGE" src="$$MAIL_IMAGEDIR$$/fm_cancel_button.gif
NAME="fm_cancel_button" nocursor border=0>
        <INPUT TYPE=HIDDEN NAME=current VALUE=$current>
        <INPUT TYPE=HIDDEN NAME=SUBMIT VALUE=$SUBMIT>
        <INPUT TYPE="IMAGE"  src="$$MAIL_IMAGEDIR$$/fm_done_button.gif
NAME="GO" border=0 nocursor>
        </td></tr></table>
        </center>
FM3

    }
   }
   else {
print <<FM3;

        <INPUT TYPE=HIDDEN NAME=current VALUE=$current>
        <INPUT TYPE=HIDDEN NAME=SUBMIT VALUE=$SUBMIT>
        <INPUT TYPE="image" src="$$MAIL_IMAGEDIR$$/empty_trash.gif
NAME="EMPTY" border=0 nocursor>
        </td></tr></table>
        </center>
FM3

    }

  }
  else {
print <<EM;

        <center>
        <table width=544 cellpadding=0 cellspacing=0 border=0
bgcolor="$FLEDE7">
        <tr><td align=center><br><br>
        <font size=4 face=Arial><i>This folder is
empty<br><br></i></font></td></tr></table>
EM
  }
}
sub funct_rename{
 $i=0;

    foreach $newfile(@newfile) {
        $old = $abs_dir."/".$oldfile[$i];
        $new = $abs_dir."/".$newfile[$i];
        rename("$abs_dir/$oldfile[$i]","$abs_dir/$newfile[$i]");
        if ($current eq 'HTML' && (-e
homedir($username)."/www/$e"."file[$i].bkup")){
        rename(homedir($      me)."/www/$oldfile[$i].bkup",
            homedir($use      e)."/www/$newfile[$i].bkup");
        open(IN, homedir($username)."/www/$newfile[$i].bkup");
        open(OUT, ">".homedir($username)."/www/$$");
        while (<IN>) {
            $line = $_ ;
            $line =~ s/destfile=.* -->/destfile=$newfile[$i] -->/;
            print OUT $line;
        }
        close(IN);
        close(OUT);
        rename(homedir($username)."/www/$$",
            homedir($username)."/www/$newfile[$i].bkup");

      }
      $i++;
   }
}
sub funct_copy(){
$i=0;
    foreach $newfile(@newfile){
      $old = $abs_dir."/".$oldfile[$i];
      $new = $abs_dir."/".$newfile[$i];
           `cp $old $new`;

      $i++;
    }
}
sub funct_move(){
  if ($current eq 'Trash'){
    open TRASH, ">>$h_dir/trash.dat" || die  "could not open trash.dat
inmove\n";
    foreach $sel_file(@select_file){
      print TRASH "$abs_dir/$sel_file\n";
    }
    close TRASH;
  }
  $current = $prev;
}
sub trash_check(
my($file) = $_[0];
$nocheck =0;
open TRSH, $h_dir."/trash.dat" ;
while(<TRSH>){
   chomp;
   $testfile = $_;
   $testfile =~ s!.*/(.*)!$1!;

     if($file eq $testfile){
       $nocheck++;
     }
  }
  close TRSH;
  if ($nocheck){
     return 0;
  }
  else{
     return 1;
  }
}
sub funct_unmove(


foreach $trfile(@trfile){
 $trhash{$trfile} = '1';
}

 open UNTRASH, ">>$h_dir/trash.dat.bak";
 open TRASH, "$h_dir/trash.dat";

   foreach $trfile(@trfile){
   while (<TRASH>){
   chomp;
   $testfile = $_;
   $testfile =~ s!.*/(.*)!$1!;
   if (!defined($trhash{$testfile})) {
   print UNTRASH $_."\n" ;
   }
  }
 }
 close UNTRASH;
 close TRASH;
 rename("$h_dir/trash.dat.bak","$h_dir/trash.dat");
}
sub print_trash(
     $count=1;
     open TRASH, "$h_dir/trash.dat" ;
       while(<TRASH>){
       $file_count++;
         chomp;
         $tr = $_ ;
         $tr =~ s!.*/(.*)!$1!;
         my $fsize = StripDecimals("/WWW/$web_dir/$tr");
#         $fsize= int $fsize;

        print qq!<td align=middle>\n<INPUT TYPE="checkbox"
NAME="TRASH_FILE" VALUE="$tr" ></td><td align=left>!;
            if($tr =~ /\.wbt$/i){
            print qq!<A
HREF="/$$MAIL_CGI_BIN$$/open_file.cgi?filename=$tr">!;
            }
            else{
            print qq!<A HREF="$web_dir/$tr">!;
            }
      print qq!<font size=2 face=arial> $tr
<i>$fsize</i></font></a></td>\n!;
      if ($count == 3){
        print "</tr><tr>\n";
      $count = 0;
      }
      $count++;
      }
}
sub move_from_email{

$current = 'Email';

# Open the database for moving the attachments.
  foreach $filepatt(@select_file){
```

FIG. 82

```
        my ($test);
        $test = 0;
        open_dbs() ;
        foreach $key ( keys %MAIL_DB ) {
         $Mail_Db->db_read($key) ;
         my $AttachmentString = $Mail_Db->Attachments() ;
         if( $AttachmentString ne "" ) {
               $AttachmentString =~ s/$filepatt[;]// ;
         cp (homedir($username)."/mail/attachments/$filepatt" ,
           homedir($username)."/www/$filepatt");
         unlink(homedir($username)."/mail/attachments/$filepatt");
         $test = 1;
           }
           $Mail_Db->Attachments($AttachmentString) ;
           $Mail_Db->db_rewrite()  ;
        }
        close_dbs() ;
          if($test != 1) {
           cp (homedir($username)."/mail/attachments/$filepatt" ,
           homedir($username)."/www/$filepatt");
         unlink(homedir($username)."/mail/attachments/$filepatt");

          }
        }
}
sub print_toolbar{

print <<TAB;
<HTML>
<TITLE>File Manager</TITLE>
<BODY BGCOLOR="$D5D1C0">

<CENTER>
TAB
if($cgiprog =~ /compose_screen.cgi/)) {
print qq!<FORM ACTION="/%%MAIL_CGI_BIN%%/compose_screen.cgi"
METHOD="POST">!;
}
if($cgiprog =~ /inbox/))
{
print qq!<FORM ACTION="/%%MAIL_CGI_BIN%%/mailbottom.cgi"
METHOD="POST">!;
}
elsif($cgiprog =~ /QuickTouch/1)) {
print qq!<FORM ACTION="/%%MAIL_CGI_BIN%%/qtDqnSdo.cgi" METHOD="POST">!;
print qq!<input type=hidden name=username value=$username>!;
print qq!<input type=hidden name=destfile value=$destfile>!;
}
else{
print qq!<FORM ACTION="/%%MAIL_CGI_BIN%%/webdesk.cgi" METHOD="POST">!;
print qq!<input type=hidden name=username value=$username>!;
}
print <<TAB;
<TABLE WIDTH=544 BORDER=0 CELLPADDING=0 CELLSPACING=0><TR><TD><TABLE
BORDER=0 CELLSPACING=0 CELLPADDING=0><TR><TD COLSPAN=7><IMG
SRC="/%%MAIL_IMAGEDIR%%/toolbar.gif" WIDTH=542 HEIGHT=6
BORDER=0></TD></TR><TR><TD ROWSPAN=2><IMG
SRC="/%%MAIL_IMAGEDIR%%/leftbar.gif" WIDTH=6 HEIGHT=61
BORDER=0></TD><TD><input type=image SRC="/%%MAIL_IMAGEDIR%%/email-
exit.gif" WIDTH=57 HEIGHT=25 BORDER=0 nocursor><INPUT TYPE="HIDDEN"


NAME="comments" VALUE="$comments"><INPUT TYPE="HIDDEN" NAME="frame"
VALUE="$frame"><INPUT TYPE="HIDDEN" NAME="subject"
VALUE="$subject"><INPUT TYPE="HIDDEN" NAME="tos" VALUE="$tos"><INPUT
TYPE="HIDDEN" NAME="ccs" VALUE="$ccs"><INPUT TYPE="HIDDEN" NAME="bccs"
VALUE="$bccs"><INPUT TYPE="HIDDEN" NAME="AttachList"
VALUE="$AttachList"><INPUT TYPE="HIDDEN" NAME="Priority"
VALUE="$Priority"><INPUT TYPE="HIDDEN" NAME="folder"
VALUE="$folder"><INPUT TYPE="HIDDEN" NAME="command"
VALUE="$command"><INPUT TYPE="HIDDEN" NAME="replyto"
VALUE="$replyto"><INPUT TYPE="HIDDEN" NAME="print_attach"
VALUE=$print_attach><INPUT TYPE="HIDDEN" NAME="actarg"
VALUE="$actarg"><INPUT TYPE="HIDDEN" NAME="CgiProg"
VALUE="$cgiprog"></td><td bgcolor=black><IMG SRC="/%%MAIL_IMAGEDIR%%/fm-
title.gif" WIDTH=413 HEIGHT=25 BORDER=0><A
HREF="http://$hostport/%%MAIL_CGI_BIN%%/helpcentral.cgi"><IMG
SRC="/%%MAIL_IMAGEDIR%%/email-help.gif" WIDTH=58 HEIGHT=25
BORDER=0></A></TD><TD ROWSPAN=2><IMG
SRC="/%%MAIL_IMAGEDIR%%/rightbar.gif" WIDTH=6 HEIGHT=61
BORDER=0></TD></TR></form><form method=POST
action=/%%MAIL_CGI_BIN%%/file_manager.cgi><TR><TD BGCOLOR=BLACK
ALIGN=MIDDLE colspan=2><input type=hidden name=current
value=$current><INPUT TYPE=IMAGE SRC="/%%MAIL_IMAGEDIR%%/$functimage"
WIDTH=98 HEIGHT=35 BORDER=0 ALT="" name=$functname nocursor>$attr<INPUT
type=image  SRC="/%%MAIL_IMAGEDIR%%/fm_upload.jpg" WIDTH=98 HEIGHT=35
BORDER=0 ALT="" name=fm_Upload nocursor><INPUT TYPE=image
SRC="/%%MAIL_IMAGEDIR%%/fm_rename.jpg" WIDTH=98 HEIGHT=35 BORDER=0
ALT="" name=fm_Rename hspace=4 nocursor><INPUT type=image
SRC="/%%MAIL_IMAGEDIR%%/fm_copy.jpg" WIDTH=98 HEIGHT=35 BORDER=0 ALT=""
name=fm_Copy nocursor><input type=hidden name=CgiProg
value=$cgiprog></TD></TR><TR><TD COLSPAN=7><IMG
SRC="/%%MAIL_IMAGEDIR%%/bottombar.gif" WIDTH=542 HEIGHT=6
BORDER=0></TD></TD></TR><tr><td colspan=7><img
src=/%%MAIL_IMAGEDIR%%/toolbar_spacer.gif
border=0></td></tr></TABLE></TD></TR></TABLE>
TAB
}
sub file_upload{
$browser = $ENV{HTTP_USER_AGENT};

if ($browser !~ /WebTV/i){
 print qq! <form method=post action=/%%MAIL_CGI_BIN%%/file_manager.cgi
enctype="multipart/form-data\" method=POST>\n !;
 standard_form() ;
 print qq! <table cellspacing=0 cellpadding=0 border=0
width=544><tr><td>!;
 print qq! <font face=arial size=3 color=#325ec5><B>Enter the file
name: </b></FONT>!;
 print qq! <input type=file name=1_attachname >\n!;
 print qq! <input type=hidden name=dir_path value=$current></td><tr><td
colspan=2>\n!;
 print qq! <center> <input type=image
src=/%%MAIL_IMAGEDIR%%/fm_upload_button.gif name=fm_upload_button
nocursor border=0 align=absmiddle vspace=10>\n!;
 print qq! <INPUT TYPE="IMAGE"
src=/%%MAIL_IMAGEDIR%%/fm_cancel_button.gif NAME="fm_cancel_button"
nocursor border=0 align=absmiddle vspace=10><br>\n!;
 if ($browser =~ /MSIE/)!
 print qq! <br><br><br><font face=arial size=3>Microsoft Internet
Explorer users: If you cannot see the "Browse" button, you need to  <a
href=http://207.68.137.46/mscownload/ieinstall/en/ffc1867.exe>download
the "HTTP Upload" patch<    from Microsoft. </FONT>!;
 }
 print qq! <input type=hidden name=current value=$prev>\n!;
 print qq! <input type=hidden name=CgiProg
value=$cgiprog></td><td></table>\n!;
 print qq! </form>!;
 print qq! </body></html>!;
   $exithere = 1;
   return($exithere);
 } else {
print <<NOT;
        </form>
       <center>
       <table width=544 cellpadding=0 cellspacing=0 border=0
bgcolor="$FLEDE7">
       <tr><td align=center><br><br><font face=Arial face=arial
size=4><i>This feature is not available for
WebTV.</font></i><br><br></td></tr></table><form Method=POST
action=/%%MAIL_CGI_BIN%%/file_manager.cgi><input type=image
src=/%%MAIL_IMAGEDIR%%/fm_continue_button.gif name=fm_continue_button
nocursor border=0>
         <input type=hidden name=current value=$prev>
         <input type=hidden name=CgiProg value=$cgiprog>
         </form>
        </center>
        </body></html>
NOT
   $exithere = 1;
   return($exithere);
  }
}


sub file_create{

  if($cgiprog =~ /QuickTouch/1){

my $Newfile = $Query->param("Newfile");
if ($Newfile eq "") {
open TMP, homedir($username)."/www/temp_html.html";
while(<TMP>){
  if (/<!-- destfile=(.*) -->/){
      $destfile = $1;
      last;
  }
}
close TMP;
print <<EOF;
<BR>
<table width=504 cellpadding=0 cellspacing=0 border=0>
<TR>
<TD><font face=arial size=3 color=#325ec5><B>Enter the file
name: </b></FONT></TD>
<TD><input type=text name=Newfile value=""></TD>
<TD COLSPAN=2 ALIGN=CENTER><INPUT TYPE=image
src=/%%MAIL_IMAGEDIR%%/fm_create_button.gif name=fm_create_button
nocursor border=0>
<INPUT TYPE="IMAGE" src=/%%MAIL_IMAGEDIR%%/fm_cancel_button.gif
NAME="fm_cancel_button" nocursor border=0>
<input type=hidden name=destfile value=$destfile>


<input type=hidden name=current value=$current>
<input type=hidden name=CgiProg
value=$cgiprog></TD></TR></TABLE></form></body></html>
EOF
}
else {
$bkup= $destfile.".bkup";
rename("$his_folder/temp_html.html" , "$his_folder/$bkup");

if($Newfile !~ /(.*).html/i){
$Newfile =~ s/$/.html/g ;
}
$destfile = $Newfile;
 open(HTM, ">".homedir($username)."/www/$Newfile");
 close(HTM);

 open NEWFILE, ">".homedir($username)."/www/temp_html.html"
                || die "can not open temp_html.html to CreateFile";
  print NEWFILE "<!-- destfile=$destfile -->\n";
  print NEWFILE "<!-- QuickTouchBody:bg:User default: -->\n" ;
  print NEWFILE "<body bgcolor=\"$D5D1C0\">\n";
  close NEWFILE;
  print "Location:
http://$hostport/%%MAIL_CGI_BIN%%/qtDqnSdo.cgi?username=$username&destfi
le=$destfile\n\n";
}

}
  else{
print <<NOT;
        </form>
       <center>
       <table width=544 cellpadding=0 cellspacing=0 border=0
bgcolor="$FLEDE7">
       <tr><td align=center><br><br><font face=Arial face=arial
size=4><i>This feature is only available from Website Design
Studio</font></i><br><br></td></tr></table><form Method=POST
action=/%%MAIL_CGI_BIN%%/file_manager.cgi><input type=image
src=/%%MAIL_IMAGEDIR%%/fm_continue_button.gif name=fm_continue_button
nocursor border=0>
         <input type=hidden name=current value=$prev>
         <input type=hidden name=CgiProg value=$cgiprog>
         </form>
        </center>
        </body></html>
NOT
  }
   $exithere = 1;
   return($exithere);

}
sub GetUploadedfile {
  NewUpload:
    {
        if ($1_attachname = $Query->param('1_attachname')) {
            $attachname = DetachFName( $Query->param('1_attachname') );
            last NewUpload if &evil_name($attachname);

            # Checks if the file exists, if yes, compares the files, if
not same
```

FIG. 8J

```
            # then saves the attachment, else does nothing.
            $attachname = CheckFile($attachment) ;

            # Clean the CGIUPLOAD temp file in the /tmp
            $CleanFName = "/tmp/CGIUPLOADtemp$$" ;
            unlink( $CleanFName ) ;

        }
    }


} # End of GetUploadedFile

sub file_edit{

      rename("$his_folder/temp_html.html","$his_folder/$destfile.bkup");
      $destfile = $select_file[0];

      if (-e "$his_folder/$destfile.bkup")
      {
        cp("$his_folder/$destfile.bkup", "$his_folder/temp_html.html");
          print "Location:
http:///$hostport/$%MAIL_CGI_BIN%%/qtDqnido.cgi?username=$username&destfi
le=&destfile\n\n";
      }
      else{
        print "Content-type: text/html\n\n <html><body
bgcolor=#D5D1CD>";
        &print_toolbar;
        print "<table width=544 border=0 cellpadding=0
cellspacing=0><tr><td bgcolor=white><br><br><center><font face=arial
size=4><i>You can not edit non-Quicktouch files using the QuickTouch
tools</i></font></center><br><br></td></tr></table>\n";

        print "<center><form METHOD=POST
action=/$%MAIL_CGI_BIN%%/file_manager.cgi><input type=hidden
name=current value=$current><input type=image
src=/$%MAIL_IMAGEDIR%%/fm_continue_button.gif name=fm_continue_button
nocursor border=0><input type=hidden name=notinhere
value=$notinhere><input type=hidden name=CgiProg
value=$cgiprog></form></center></body></html>\n";
   $exithere = 1;
   return($exithere);

      }
}

sub file_empty{
     open TRASH, "$h_dir/trash.dat" ;
     while(<TRASH>){
     chomp;
       unlink($_);
       }
     close(TRASH);
    `rm $h_dir/trash.dat`;
}
sub move_from_web{

foreach $sel(@select_file){
      cp (homedir($username)."/www/$sel"


 ,homedir($username)."/mail/attachments/$sel");
       unlink(homedir($username)."/www/$sel");
    }
 }

 sub PrintError {

  open( TMPFM, ">>/tmp/fmem.log" ) ;
  print TMPFM @_ ;
  print TMPFM "\n ------ \n" ;
  close(TMPFM) ;
  chmod 0666, "/tmp/fmem.log" ;

 } # End of PrintError

 1;
```

```
  #!%%PERL%%
  # --*- Perl -*-

  # $Header: /proj/file-manager/CGI/file_manager.xcgi.v 1.19 1997/11/05
  17:35:41 ramman Exp $
  use AddINC qw(%%PERL_LIB%%);

  use vars qw($frame $bgcolor $Query $current $prev $services);
  require 'common_cgi.pl';
  CheckLogin();

  require 'file_manager.pl';

  &file_manager_setup();

  $frame = $Query->param('frame');
  $fm_upload_button = $Query->param('fm_upload_button.x') ;
  $prev = $Query->param('prev') ;
  $fm_attach = $Query->param('fm_Attach.x');
  $fm_rename = $Query->param('fm_Rename.x');
  $fm_create = $Query->param('fm_Create.x');
  $fm_create_button = $Query->param('fm_create_button.x');
  $fm_copy = $Query->param('fm_Copy.x');
  $fm_upload = $Query->param('fm_Upload.x');
  $fm_continue_button = $Query->param('fm_continue_button.x');
  $fm_cancel_button = $Query->param('fm_cancel_button.x');
  $fm_upload_button = $Query->param('fm_upload_button.x');
  $fm_edit = $Query->param('fm_Edit.x');
  $fm_done_button = $Query->param('fm_done_button.x');
  $empty = $Query->param('EMPTY.x');
  $go = $Query->param('GO.x');
  $current = $Query->param('current');
  @select_file = $Query->param('FILE');
  $cgiprog = $Query->param('CgiProg');
  my($h_dir) = main::homedir($main::USERNAME);
  $services = QueryUserAccounts($Query, $username, 'service');
  if ($services eq "") {
      $services = "EIWIM";
  }
  if($fm_upload){
       if($services =~ /S/i){
      print "Location:
http:///$hostport/$%MAIL_CGI_BIN%%/productinfo_wsftwd.cgi\n\n";
   }
   }
   if($fm_create){
   if($services !~ /MIB/i){
      print "Location:
http://$hostport/$%MAIL_CGI_BIN%%/productinfo_wsftwd.cgi\n\n";
   }
   }
  if ($fm_upload_button){
     $Submit = 'Upload';
     }
  if ($fm_create_button){
     &file_create;
     }
  if ($cgiprog =~ /compose/ || $cgiprog =~ /inbox/){
  $functimage = "fm_attach.jpg";
  $functname = "fm_Attach";

  }
  else{
  $functimage = "fm_edit.jpg";
  $functname = "fm_Edit";
  }

  if ($SUBMIT1 =~ /Create/){
      &file_create;
      }
  if($fm_edit && $select_file[0] && ($cgiprog eq 'QuickTouch' || $cgiprog
  eq 'WebDesk')) {
          &file_edit;
  }
  if($fm_attach eq "" && $fm_edit eq "" && $fm_upload  eq "" && $fm_create
  eq "" && $fm_rename eq "" && $go eq "" && $fm_copy eq "" && $empty eq ""
  && $fm_create_button eq '' && $fm_upload_button eq '' &&
  $fm_done_button eq "" && $fm_continue_button eq "" && $fm_cancel_button
  eq ""){
   @names = $Query->param;
   foreach $test(@names) {
       ($var,$junk)=split(/\./,$test);
       if ($junk =~ m/"m/i,) {
            $current=$var;
       }
   }
  }
  if($empty){
  &file_empty;
  }

  if( $Submit eq "Upload" ) {
    &GetUploadedFile ;
  }

   if ($current eq 'Trash') {
     @selected = $Query->param('FILE');
     open (TRASH, ">>/tmp/trash");
     print TRASH "@selected";
   }
   if (!defined($fm_attach)){
       phead('$Header: /proj/file-manager/CGI/file_manager.xcgi,v 1.19
  1997/11/05 17:35:41 ramman Exp $ ');
       print("<HEAD><TITLE>File Manager</TITLE></HEAD>\n");
       &body($bgcolor);
       &file_manager($current);
       print("</BODY>\n</HTML>\n");
       goto EXITHERE;
   }
   elsif((($cgiprog =~ /compose_screen.cgi/ || $cgiprog =~ /inbox/))){
       foreach (@select_file ) {
          $SelString = $SelString."\c_".$_ ;
       }
      print"Location:
  http:///$hostport/$%MAIL_CGI_BIN%%/compose_screen.cgi?formcommand=OpenFor
  m&dir_from=$current&frame=$frame&username=$USERNAME&file_attach=$SelStr
  ing &msg\n\n";
   }
  else{
         print "Content-type: text/html\n\n";
         &print_toolbar;
```

FIG. 8d

```
        <center>
        <table width=544 cellpadding=0 cel'    ing=0 border=0
bgcolor="#F1EDE7">
        <tr><td align=center><br><br>
        <H3><font face=Arial color=blue>This Feature  is only Available
for Email users</font></h3></td></tr></table>
          <form method=POST action=/%%MAIL_CGI_BIN%%/file_manager.cgi>
          <input type=image src=/%%MAIL_IMAGEDIR%%/fm_continue_button.gif
name=fm_continue_button>
          <input type=hidden name=current value=$mail>
          </form>
          </center>
          </body></html>

OT
EXITHERE:
}
#!%%PERL%%

use AddINC qw(%%PERL_LIB%%);

require 'common_cgi.pl';

CheckLogin();
use vars qw($Query $current $attachname $add_photo $c $card $type $frame
$username $USERNAME $PROTODIR
        $preq_view $prog_edit $attach_dir $bgcolor $hostport);

require 'pim-convert.pl';
require 'file_manager.pl';

$current = "Email";
$fm_upload_button = $Query->param('fm_upload_button.x') ;
$fm_cancel_button = $Query->param('fm_cancel_button.x');
$import_pim = $Query->param('ImportPim.x');
$export_pim = $Query->param('ExportPim.x');
$convert_pim = $Query->param('ConvertPim.x');
$import_file = $Query->param('ImportFile');
if( $import_pim ) {
    print "Content-type:text/html\n\n";
    &body($bgcolor);
    print_tool_bar();
    print "<CENTER>";
    file_upload();
} elsif ( $export_pim ) {
    ##########################################
    ## TODO: Call an export routine
    ##########################################
}

if( $fm_upload_button ) {
    $import_file = &GetUploadedFile ;
    print "Content-type:text/html\n\n";
    &body($bgcolor);
    print_tool_bar();
    $file_ok = check_file($import_file);
    if ($file_ok) {
      get_header($import_file);
    } else {
      print_bad_file($import_file);
    }
}
if ($convert_pim) {
    convert_pim($import_file);
}
# -*- perl -*-

use CGI;
use AddINC qw(%%PERL_LIB%%);
use CommonDb;
use DbPim;
## use strict;
## use vars qw($Database_Opened_Pim $Pim_Db_Obj %PIM_DB $Pim_Db
@contacts %PIMS);
use vars qw($Query $current $homedir $attach_dir $Pim_Db %PIM_DB);

sub check_file {
    my $num_records = 0;
    my $homedir = homedir($USERNAME);
    my $ATTACHDIR = $homedir."/".$attach_dir ;
    if (! -T "$ATTACHDIR/@_") {
      return(0);
    }
    open (PIM,"<$ATTACHDIR/@_") || die "Could not open import file :
$!\n";
    while(<PIM>) {
        @records = split (",",$_);
      if (! $num_records) {
          if ($#records < 1) {
            next;
          } else {
            $num_records = $#records;
          }
      } else {
          if ($num_records == $#records) {
            return(1);
          } else {
            $num_records = $#records;
          }
      }
    }
    return(0);
}
sub print_bad_file {
    print qq!<CENTER><FONT FACE=ARIAL SIZE=4><I>$import_file is not a
valid import file.<BR>Files should be comma delimited text files only.!;
    print qq!<br>Please try again.</I></FONT></CENTER>!;
}
sub get_header {
    $homedir = homedir($USERNAME);
    $ATTACHDIR = $homedir."/".$attach_dir ;

    open (PIM,"<$ATTACHDIR/@_") || die "Could not open import file :
$!\n";

    my ($addr_exist,$db_exist,$who,$card);
    srand( time() ^ ($$ + ($$ << 15)) );
    $got_header = 0;
    while (<PIM>) {

      chop $_;
        ## Remove quotes
        $_ =~ s/^\s*(\S)/$1/;
        $_ =~ s/("")|("")|("$)|("$//g;
        $_ =~ s/(("),("))|(("),("))/,/g;

        @records = split (",",$_);

        if ( ($#records >= 1) && (!$got_header) ) {
            foreach $index (0...$#records) {
                $fields[$index] = $Query->param('FIELD'.$index);
            }
            $got_header = 1;
            next;
        }

        if ($got_header) {
            $card=rand 9999999999;

            $Pim_Db->FirstName($records[$fields[0]]);
            $Pim_Db->LastName($records[$fields[1]]);
            $Pim_Db->Title($records[$fields[2]]);
            $Pim_Db->Company($records[$fields[3]]);
            $Pim_Db->Address1($records[$fields[4]]);
            $Pim_Db->City1($records[$fields[5]]);
            $Pim_Db->State1($records[$fields[6]]);
            $Pim_Db->Zip1($records[$fields[7]]);
            $Pim_Db->Address2($records[$fields[8]]);
            $Pim_Db->City2($records[$fields[9]]);
            $Pim_Db->State2($records[$fields[10]]);
            $Pim_Db->Zip2($records[$fields[11]]);
            $Pim_Db->Email($records[$fields[12]]);
            $Pim_Db->Url($records[$fields[13]]);
            $Pim_Db->Phone($records[$fields[14]]);
            $Pim_Db->Phone2($records[$fields[15]]);
            $Pim_Db->Phone3($records[$fields[16]]);
            $Pim_Db->Fax($records[$fields[17]]);
            $Pim_Db->Fax2($records[$fields[18]]);
            $Pim_Db->Pager($records[$fields[19]]);
            $Pim_Db->Mobile($records[$fields[20]]);
            $Pim_Db->Notes($records[$fields[21]]);

            $Pim_Db->db_write($card);
        }
    }
    close_dbs();
    print "Content-type:text/html\n\n";
    &body($bgcolor);
    print_tool_bar();
    print qq!<CENTER><FONT FACE=ARIAL SIZE=4>Contact Manager Import Is
Complete!</FONT></CENTER>!;
}

sub field_selection {
    my $field_num = $_[1];
    print qq!<TR><TD WIDTH=160><B>$_[0] </B></TD><TD WIDTH=34
ALIGN=LEFT></TD>!;
    print qq!<TD WIDTH=360 ALIGN=LEFT><SELECT NAME=FIELD$field_num>!;
    print qq!<OPTION VALUE="0">No Import!;
    foreach $index (1 ...$#records+1) {
      print qq!<OPTION VALUE="$index">$records[$index-1]!;
    }

      $_ =~ s/(\S)\s*$/$1/;
      $_ =~ s/("")|("")|("$)|("$)//g;
        $_ =~ s/(("),("))|(("),("))/,/g;

        @records = split (",",$_);

        if ( ($#records >= 1) && (!$got_header) ) {
        ## TODO: Fill in
          print qq!<FORM ACTION="/%%MAIL_CGI_BIN%%/pim-convert.cgi"
METHOD=POST>!;
          print qq!<CENTER><TABLE WIDTH=544 BORDER=0 CELLPADDING=0
CELLSPACING=0>!;
          field_selection("First Name",1);
          field_selection("Last Name",2);
          field_selection("Title",3);
          field_selection("Company",4);
          field_selection("Business Address",5);
          field_selection("Business City",6);
          field_selection("Business State",7);
          field_selection("Business Zip",8);
          field_selection("Home Address",9);
          field_selection("Home City",10);
          field_selection("Home State",11);
          field_selection("Home Zip",12);
          field_selection("Email",13);
          field_selection("URL",14);
          field_selection("Business Phone",15);
          field_selection("Home Phone",16);
          field_selection("Business Phone 2",17);
          field_selection("Fax",18);
          field_selection("Fax 2",19);
          field_selection("Pager",20);
          field_selection("Mobile",21);
          field_selection("Notes",22);
          print qq!</TABLE>!;
          $got_header = 1;
        }
    }
    print qq!<INPUT TYPE=HIDDEN NAME="ImportFile" VALUE="@_">!;
    print qq!<INPUT TYPE=IMAGE SRC="/%%MAIL_IMAGEDIR%%/" BORDER=0
NAME="ConvertPim"></FORM>!;
}

sub convert_pim {
    $homedir = homedir($USERNAME);
    $ATTACHDIR = $homedir."/".$attach_dir ;

    open (PIM,"<$ATTACHDIR/@_") || die "Could not open import file :
$!\n";

    my ($addr_exist,$db_exist,$who,$card);

    open_dbs_pim();
    srand( time() ^ ($$ + ($$ << 15)) );
    $got_header = 0;

    while (<PIM>) {
        chop $_;
        ## Remove quotes
        $_ =~ s/^\s*(\S)/$1/;

        print qq!</SELECT></TD></TR>!;
    }
```

FIG. 85

PCT/US98/16894
WO 99/09658
27 / 36

```
package upgrade;
use AddINC qw(%%PERL_LIB%%);
use strict;
use Business::CreditCard;
use billing qw(QueryBillRecord ModifyBillRecord);
use useraccounts qw(@SERVICE_SEP userlogin QueryUserAccounts ModifyUserAccounts);
use reghistory qw(AddHistory);
use action qw(GenNewActionNumber AddAction);
use rutils  qw(fo live rf dn dnstr @DELIM);
use vars qw(@ISA @EXPORT_OK);
use Exporter();
@ISA = qw(Exporter);
@EXPORT_OK = qw(upgrade);

sub upgrade {
   my ($cgi, %params) = @_;
   my ($k, $v, %pparams, $prices, $m, $dp, $d, $p, @currentproducts, $tmp, $bot, $up, $new,
$username, @upgrades, @upgradeopts, @upgrades, @services, @services, @currentproducts, @upgr
ade, $service, %upe, %buf, $new, @new);
   $username = @params{username} || @params{'username'} || @params{'__username__'};
# Uncomment when handling different resellers differently.
$if ($cgi->server_name =~ /eony/i) {
   if (@params{'upgrade.x'} or @params{'change.x'} or @params{'continueorder.x'}) {
      @params{billno} = QueryUserAccounts{@cgi, $username, 'billno'};
      if (@params{'change.x'}) {
         my $html = fo("%%HTML_INST%%/changeinfo.html");
         if (@params{'password-old'} ne '' and defined(@params{'password-old'})) {
            if (userlogin(@cgi, $username, @params{'password-old'})) {
               if (@params{password1} eq @params{password2}) {
                  @params{password} = @params{password1} || @params{'password-old'};
                  ModifyUserAccounts(@cgi, $username, \%params);
                  ModifyBillRecord(@cgi, QueryUserAccounts{@cgi, $username, 'billno'}, \%par
ams);
               } else {
                  @params{'password-old'} = '';
                  @params{password1} = '';
                  @params{password2} = '';
                  $html = rf(\%params, $html);
                  $html =~ s/<!--%%ml-holder%%-->/Please retype your new passwords carefully
!/;
                  print "Content-type:text/html\n\n${html}\n"; exit;
               }
            } else {
               @params{'password-old'} = '';
               @params{password1} = '';
               @params{password2} = '';
               $html = rf(\%params, $html);
               $html =~ s/<!--%%ml-holder%%-->/Your password is incorrect!/;
               print "Content-type:text/html\n\n${html}\n"; exit;
            }
         } else {
            @params{'password-old'} = '';
            @params{password1} = '';
            @params{password2} = '';
            $html = rf(\%params, $html);
            $html =~ s/<!--%%ml-holder%%-->/Please supply your old password!/;
            print "Content-type:text/html\n\n${html}\n"; exit;
         }
      }
      @params{user} = $username;
      @params{billno} = QueryUserAccounts{@cgi, $username, 'billno'};
      @params{password} = QueryUserAccounts{@cgi, $username, 'password'};
      @params{currentproducts} = QueryUserAccounts{@cgi, $username, 'service'};
      @params{currentproducts} =~ s/BONT-101/8/;
      @currentproducts = @params{currentproducts};
      @params{'saint-service'} = @params{currentproducts};
      @params{currentproducts} = '';
      @params{firstname} = QueryBillRecord{@cgi, @params{billno}, 'firstname'};
      @params{lastname} = QueryBillRecord{@cgi, @params{billno}, 'lastname'};
      @params{company} = QueryBillRecord{@cgi, @params{billno}, 'company'};
      @params{address1} = QueryBillRecord{@cgi, @params{billno}, 'address1'};
      @params{address2} = QueryBillRecord{@cgi, @params{billno}, 'address2'};
      @params{city} = QueryBillRecord{@cgi, @params{billno}, 'city'};
      @params{state} = QueryBillRecord{@cgi, @params{billno}, 'state'};
      @params{postcode} = QueryBillRecord{@cgi, @params{billno}, 'postcode'};
      @params{phone} = QueryBillRecord{@cgi, @params{billno}, 'phone'};
      @params{email} = QueryBillRecord{@cgi, @params{billno}, 'email'};
      @params{cctype} = QueryBillRecord{@cgi, @params{billno}, 'cctype'};
      @params{ccno} = QueryBillRecord{@cgi, @params{billno}, 'ccno'};
      @params{ccexp} = QueryBillRecord{@cgi, @params{billno}, 'ccexp'};
      @params{ccname} = QueryBillRecord{@cgi, @params{billno}, 'ccname'};
      @params{billcycle} = QueryBillRecord{@cgi, @params{billno}, 'billcy'   '};
      @params{ccbille1} = QueryBillRecord{@cgi, @params{billno}, 'ccbille1'};
      @params{ccbille2} = QueryBillRecord{@cgi, @params{billno}, 'ccbille2'};
      @params{cccity} = QueryBillRecord{@cgi, @params{billno}, 'cccity'};
      @params{ccstate} = QueryBillRecord{@cgi, @params{billno}, 'ccstate'};
      @params{ccpostcode} = QueryBillRecord{@cgi, @params{billno}, 'ccpostcode'};
      @params{new} = QueryBillRecord{@cgi, @params{billno}, 'new'};
      @params{upgrade} = QueryBillRecord{@cgi, @params{billno}, 'upgrade'};
      if (defined(@params{new}) and @params{new} ne '') {
         @new = split(/\|/, @params{new});
         @params{new} = undef;
         foreach (@new) { @params{"new-${_}"} = $_ }
      }
      if (@currentproducts =~ /\|/) {
         @currentproducts = split(/\|/, @currentproducts)
      } else {
         push @currentproducts, (@currentproducts)
      }
      @prices = fo("%%DATA_INST%%/price.dat");
      @upgrades = fo("%%DATA_INST%%/upgrade.dat");
      %upe = ('universal', 'universal');
      my %buf;
      foreach (@currentproducts) {
         @prices =~ /@{$_}->{.}*/;
         $dp = $&; $dp =~ /@{$_}->/; $dp = $';
         ($d, $p) = split(/:/, $dp);
         @params{currentproducts} .= "$d, ";
         if (@upgrades =~ /@{$_}->{.}*/) {
            @upgradeopts = $&;
            @upgradeopts =~ /@{$_}->/;
            @upgradeopts = $';
            (@upgradeopts, @services) = split(/\)\(/, @upgradeopts);
            @upgradeopts =~ s/^\(//;
            @services =~ s/\)$//;
            if (@upgradeopts =~ /\|/) { @upgrades = split(/\|/, @upgradeopts) }
            else { push @upgrades, (@upgradeopts) }
            if (@services =~ /\|/) { @services = split(/\|/, @services) }
            else { push @services, (@services) }
            unless (@upe{universal} ne 'universal') {
               if (@#upgrades >= 0) {
                  foreach $upgrade (@upgrades) {
                     $upe{$upgrade} = $upgrade
                  }
               }
               $upe{universal} = '';
```

FIG. 86

upgradejxpm

```
        } else {
          if (@@upgrades >= 0) {
            foreach $upgrade (@upgrades) {
              if (%ups{$upgrade} eq $upgrade) { $buf{$upgrade} = $upgrade }
            }
          }
          $ups = \%buf;
          %buf = ();
          $ups{universal} = '';
        }
        foreach $service (@services) { $new{$service} = $service }
      }
      foreach (@currentproducts) { $ups{$_} = ''; $new{$_} = ''; }
    }
    $pparams{currentproducts} =~ s/,\s0//;
    $top = fo("%%HTML_INST%%/upgradeltop.html");
    my $action = $cgi->script_name;
    $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
    $top = rf( \%pparams, $top);
    $bot = fo("%%HTML_INST%%/upgradelbot.html");
    $bot = rf( \%pparams, $bot);
    $up = rf( \%pparams, UPS_HTML($ups));
    $new = rf(\%pparams, NEW_HTML($new));
    print "Content-type:text/html\n\n$[top]\n$[up]\n$[new]\n$[bot]\n";
    exit;
  } elsif ($pparams{'submit.x'}) {
    $pparams{service} = QueryUserAccounts($cgi, $username, 'service');
    $pparams{service} =~ s/SONIT-101/S/;
    my ($u, $n);
    ($pparams{currentproducts}, $u, $n) = UPS_NEW($pparams{service});
    $pparams{user} = $username;
    $pparams{billno} = QueryUserAccounts($cgi, $username, 'billno');
    $pparams{password} = QueryUserAccounts($cgi, $username, 'password');
    $pparams{firstname} = QueryBillRecord($cgi, $pparams{billno}, 'firstname');
    $pparams{lastname} = QueryBillRecord($cgi, $pparams{billno}, 'lastname');
    $pparams{company} = QueryBillRecord($cgi, $pparams{billno}, 'company');
    $pparams{address1} = QueryBillRecord($cgi, $pparams{billno}, 'address1');
    $pparams{address2} = QueryBillRecord($cgi, $pparams{billno}, 'address2');
    $pparams{city} = QueryBillRecord($cgi, $pparams{billno}, 'city');
    $pparams{state} = QueryBillRecord($cgi, $pparams{billno}, 'state');
    $pparams{postcode} = QueryBillRecord($cgi, $pparams{billno}, 'postcode');
    $pparams{phone} = QueryBillRecord($cgi, $pparams{billno}, 'phone');
    $pparams{email} = QueryBillRecord($cgi, $pparams{billno}, 'email');
    $prices = fo("%%DATA_INST%%/price.dat");
    while (($k, $v) = each %pparams) {
      if ($k =~ /new/i and $v ne '' and defined($v)) {
        $pparams{new} .= "$v|";
        push @new, ($v);
        $prices =~ /$[v]->[.]*/;
        $dp = $&; $dp =~ /$[v]->/; $dp = $';
        ($d) = split(/:/, $dp);
        $n .= "<LI>$d";
      }
    }
    if ($pparams{new}) { $pparams{new} =~ s/\|$//; }
    if ($pparams{upgrade} ne '' && defined($pparams{upgrade})) {
      $pparams{service} =~ s/|$PKB|/$pparams{upgrade}/;
      $prices =~ /$pparams{upgrade}->[.]*/;
      $dp = $&; $dp =~ /$[v]->/; $dp = $';
      ($d) = split(/:/, $dp);
      $n .= "<LI>$d";
    }
    push @new, split(/\|/, $pparams{service});
    $pparams{service} = join($SERVICE_SEP, @new);
    $pparams{octype}  = $params{octype} || $params{'octype'};
    $pparams{ccno} = $params{ccno} || $params{'ccnot'};
    $pparams{ccname} = $params{ccname} || $params{'ccname'} || "$pparams{first
ame}$lastname}";
    $pparams{ccexp} = $params{ccexp} || $params{'ccexp'};
    if ($pparams{octype} =~ /^C$/i or $pparams{octype} =~ /^CO$/i) {
      $pparams{billcycle} = 'A';
    } else {
      $pparams{billcycle} = $params{billcycle} || $params{'billcycle'};
    }
    $pparams{ccbille1} = $params{ccbille1} || $params{'ccbille1'} || $pparams{
    $pparams{ccbille2} = $params{ccbille2} || $params{'ccbille2'} || $pparams{
    $pparams{cccity} = $params{cccity} || $params{'cccity'} || $pparams{city};
    $pparams{ccstate} = $params{ccstate} || $params{'ccstate'} || $pparams{sta
    $pparams{ccpostcode} = $params{ccpostcode} || $params{'ccpostcode'} || $pp
ode};
    $pparams{upgrade} = $params{upgrade} || $params{'upgrade'};
    ModifyBillRecord($cgi, QueryUserAccounts($cgi, $username, 'billno'), '   pr
    unless ($pparams{new} or $params{upgrade}) {
      $top = fo("%%HTML_INST%%/upgradeltop.html");
      $top =~ s/<!--$nl-holder$-->/You have not selected anything for purchase
!/;
      my $action = $cgi->script_name;
      $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
      $top = rf( \%pparams, $top);
      $bot = fo("%%HTML_INST%%/upgradelbot.html");
      $bot = rf( \%pparams, $bot);
      $up = rf( \%pparams, UPS_HTML(%%u));
      $new = rf(\%pparams, NEW_HTML(%%n));
      print "Content-type:text/html\n\n$[top]\n$[up]\n$[new]\n$[bot]\n";
      exit;
    } elsif ($pparams{octype} eq '' or !defined($pparams{octype})) {
      $top = fo("%%HTML_INST%%/upgradeltop.html");
      $top =~ s/<!--$nl-holder$-->/Please select a payment option!/;
      my $action = $cgi->script_name;
      $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
      $top = rf( \%pparams, $top);
      $bot = fo("%%HTML_INST%%/upgradelbot.html");
      $bot = rf( \%pparams, $bot);
      $bot =~ s/<!--$octype-holder$-->/<font color=darkred>/;
      $bot =~ s/<!--$octype-end$-->/<\/font>/;
      $up = rf( \%pparams, UPS_HTML(%%u));
      $new = rf(\%pparams, NEW_HTML(%%n));
      print "Content-type:text/html\n\n$[top]\n$[up]\n$[new]\n$[bot]\n";
      exit;
    } else {
      if ($pparams{ccno} eq '' or !defined($pparams{ccno}) or
          $pparams{ccname} eq '' or !defined($pparams{ccname}) or
          $pparams{ccexp} eq '' or !defined($pparams{ccexp})) {
        if ($pparams{octype} =~ /|VMA|/i) {
          $top = fo("%%HTML_INST%%/upgradeltop.html");
          $top =~ s/<!--$nl-holder$-->/Please make sure you properly supplie
t card number, name on card, and\/or expiration date!/;
          my $action = $cgi->script_name;
          $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
          $top = rf( \%pparams, $top);
          $bot = fo("%%HTML_INST%%/upgradelbot.html");
          $bot = rf( \%pparams, $bot);
          unless ($pparams{ccno}) {
            $bot =~ s/<!--$ccno-holder$-->/<font color=darkred>/;
            $bot =~ s/<!--$ccno-end$-->/<\/font>/;
          }
```

FIG. 87

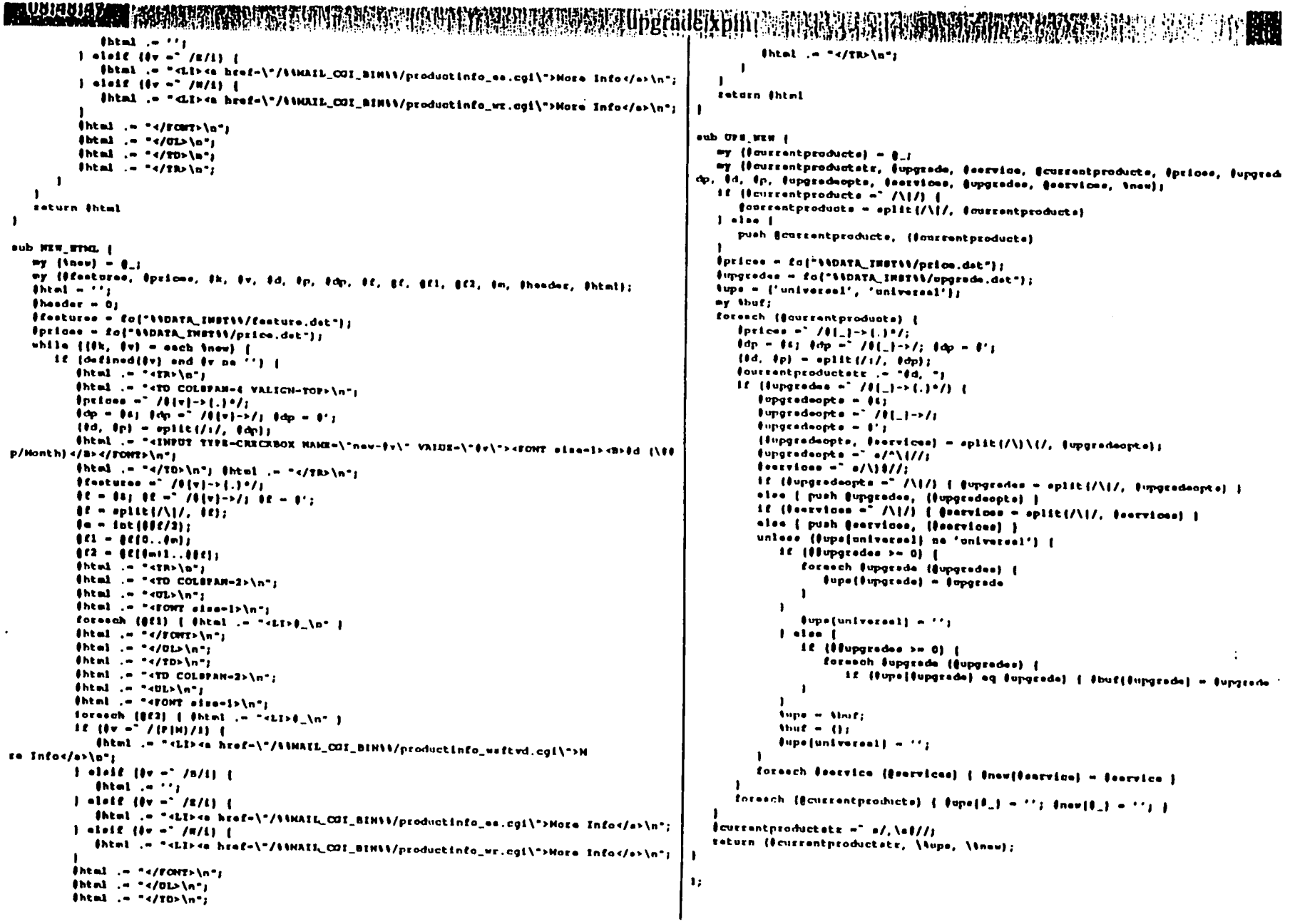
upgrade.xpm

```
            unless (@pperams{cnsms}) {
               $bot =~ s/<!--@cnsms-holder@-->/<font color=darkred>/;
               $bot =~ s/<!--@cnsms-end@-->/<\/font>/;
            }
            unless (@pperams{ccexp}) {
               $bot =~ s/<!--@ccexp-holder@-->/<font color=darkred>/;
               $bot =~ s/<!--@ccexp-end@-->/<\/font>/;
            }
            $up = rf( \%pperams, UPS_HTML{%%u});
            $new = rf(\%pperams, NEW_HTML{%%n});
            print "Content-type:text/html\n\n%{top}\n%{up}\n%{new}\n%{bot}\n";
            exit;
         }
      }

      unless (@pperams{billcycle}) {
         $top = fo("%%HTML_INST%%/upgradeltop.html");
         $top =~ s/<!--@ml-holder@-->/Please select the billing cycle you desired!/;
         my $action = $cgi->script_name;
         $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
         $top = rf( \%pperams, $top);
         $bot = fo("%%HTML_INST%%/upgradelbot.html");
         $bot = rf( \%pperams, $bot);
         $bot =~ s/<!--@billcycle-holder@-->/<font color=darkred>/;
         $bot =~ s/<!--@billcycle-end@-->/<\/font>/;
         $up = rf( \%pperams, UPS_HTML{%%u});
         $new = rf(\%pperams, NEW_HTML{%%n});
         print "Content-type:text/html\n\n%{top}\n%{up}\n%{new}\n%{bot}\n";
         exit;
      }

      if (@pperams{octype} =~ /[AVM]/i) {
         if (validate(@pperams{ccno}) == 0) {
            $top = fo("%%HTML_INST%%/upgradeltop.html");
            $top =~ s/<!--@ml-holder@-->/Please reenter your credit card number carefully
!/;
            my $action = $cgi->script_name;
            $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
            $top = rf( \%pperams, $top);
            $bot = fo("%%HTML_INST%%/upgradelbot.html");
            $bot = rf( \%pperams, $bot);
            $bot =~ s/<!--@ccno-holder@-->/<font color=darkred>/;
            $bot =~ s/<!--@ccno-end@-->/<\/font>/;
            $up = rf( \%pperams, UPS_HTML{%%u});
            $new = rf(\%pperams, NEW_HTML{%%n});
            print "Content-type:text/html\n\n%{top}\n%{up}\n%{new}\n%{bot}\n";
            exit;
         }

         my $ccexp = $pperams{ccexp};
         if ($ccexp !~ /\d\d?\/\d\d?\/\d\d/) {
            $top = fo("%%HTML_INST%%/upgradeltop.html");
            $top =~ s/<!--@ml-holder@-->/Please reenter your credit card expiration date
in format mm\/dd\/yy/;
            my $action = $cgi->script_name;
            $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
            $top = rf( \%pperams, $top);
            $bot = fo("%%HTML_INST%%/upgradelbot.html");
            $bot = rf( \%pperams, $bot);
            $bot =~ s/<!--@ccexp-holder@-->/<font color=darkred>/;
            $bot =~ s/<!--@ccexp-end@-->/<\/font>/;
            $up = rf( \%pperams, UPS_HTML{%%u});
            $new = rf(\%pperams, NEW_HTML{%%n});
            print "Content-type:text/html\n\n%{top}\n%{up}\n%{new}\n%{bot}\n";
            exit;
         }

         my ($ccexpmm, $ccexpdd, $ccexpyy) = split(/\//, $ccexp);
         if ($ccexpmm < 10) {
            if ($ccexpmm =~ /^\d$/) { $ccexpmm = "0$ccexpmm" }
         }
         if ($ccexpdd < 10) {
            if ($ccexpdd =~ /^\d$/) { $ccexpdd = "0$ccexpdd" }
         }
         my ($new) = split(/\s/, dnstr(dn()));
         my ($ew, $dd, $yy) = split(/\//, $new);
         if ("$yy$ew$dd" > "19$ccexpyy$ccexpmm$ccexpdd") {
            $top = fo("%%HTML_INST%%/upgradeltop.html");
            $top =~ s/<!--@ml-holder@-->/Please reenter your credit card expi
r/;
            my $action = $cgi->script_name;
            $top =~ s/<FORM>/<FORM METHOD="POST" ACTION="$action">/i;
            $top = rf( \%pperams, $top);
            $bot = fo("%%HTML_INST%%/upgradelbot.html");
            $bot = rf( \%pperams, $bot);
            $bot =~ s/<!--@ccexp-holder@-->/<font color=darkred>/;
            $bot =~ s/<!--@ccexp-end@-->/<\/font>/;
            $up = rf( \%pperams, UPS_HTML{%%u});
            $new = rf(\%pperams, NEW_HTML{%%n});
            print "Content-type:text/html\n\n%{top}\n%{up}\n%{new}\n%{bot}\n";
            exit;
         }
      }
   }
   $top = fo("%%HTML_INST%%/thankyoultop.html");
   $bot = fo("%%HTML_INST%%/thankyoulbot.html");
   $bot = rf( { service => $pperams{service} }, $bot);
   print "Content-type:text/html\n\n%{top}<TR><TD colspan=4><UL><FONT size=+1 c
lue>${n}</FONT></UL></TD></TR>%{bot}\n";
   exit;
} elsif ($params{'accept.x'}) {
   $params{billno} = QueryUserAccounts($cgi, $username, 'billno');
   $pperams{octype}  = QueryBillRecord($cgi, $params{billno}, 'octype');
   $pperams{ccno} = QueryBillRecord($cgi, $params{billno}, 'ccno');
   $pperams{ccname} = QueryBillRecord($cgi, $params{billno}, 'ccname');
   $pperams{ccexp} = QueryBillRecord($cgi, $params{billno}, 'ccexp');  (
   $pperams{billcycle} = QueryBillRecord($cgi, $params{billno}, 'billcycl- );
   $pperams{ccbilla1} = QueryBillRecord($cgi, $params{billno}, 'ccbilla1');
   $pperams{ccbilla2} = QueryBillRecord($cgi, $params{billno}, 'ccbilla2');
   $pperams{cccity} = QueryBillRecord($cgi, $params{billno}, 'cccity');
   $pperams{ccstate} = QueryBillRecord($cgi, $params{billno}, 'ccstate');
   $pperams{ccpostcode} = QueryBillRecord($cgi, $params{billno}, 'ccpostcode');
   $pperams{new} = QueryBillRecord($cgi, $params{billno}, 'new');
   $pperams{upgrade} = QueryBillRecord($cgi, $params{billno}, 'upgrade');
   $pperams{service} = QueryUserAccounts($cgi, $username, 'service');
   my $oldservice = $pperams{service};
   if ($pperams{upgrade}) {
      $pperams{service} =~ s/[PPNB]/$pperams{upgrade}/i;
   }
   if ($pperams{new}) {
      $pperams{service} .= "|$pperams{new}";
   }

   $pperams{service} =~ s/\|+/\|/g;
   $pperams{service} =~ s/^\|//;
   $pperams{service} =~ s/\|$//;
```

FIG. 88

```
    ModifyUserAccounts(@cgi, $username, { service => $pparams{service} });

    my $action;
    if ($pparams{upgrade}) {
       $action = GenNewActionNumber(@cgi);
       $pparams{service} = $pparams{upgrade};
       $pparams{transaction} = 'U';
       AddAction(@cgi, $action, \%pparams);
       AddHistory(@cgi, $params{billno}, $action);
       $action = GenNewActionNumber(@cgi);
       AddAction(@cgi, $action, { service=>$oldservice, transaction=>'D' });
       AddHistory(@cgi, $params{billno}, $action);
    }
    if ($pparams{new}) {
       @new = split(/\|/, $pparams{new});
       foreach (@new) {
          $action = GenNewActionNumber(@cgi);
          $pparams{service} = $_;
          $pparams{transaction} = 'p';
          AddAction(@cgi, $action, \%pparams);
          AddHistory(@cgi, $params{billno}, $action);
       }
    }

    $params{new} = '';
    $params{upgrade} = '';
    $params{cctype} = '';
    $params{ccno} = '';
    $params{ccname} = '';
    $params{ccexp} = '';
    $params{billcycle} = '';
    $params{ccbilla1} = '';
    $params{ccbilla2} = '';
    $params{cccity} = '';
    $params{ccstate} = '';
    $params{ccpostcode} = '';

    ModifyBillRecord(@cgi, $params{billno}, \%params);
    system("./webdesk.cgi");
} elsif ($params{'exittowebdesk.x'}) {
    system("./webdesk.cgi");
} elsif ($params{'areyousure.x'}) {
    $pparams{cctype}  = $params{cctype} || $params{'cctype'};
    $pparams{ccno} = $params{ccno} || $params{'ccno'};
    $pparams{ccname} = $params{ccname} || $params{'ccname'};
    $pparams{ccexp} = $params{ccexp} || $params{'ccexp'};
    $pparams{billcycle} = $params{billcycle} || $params{'billcycle'};
    $pparams{ccbilla1} = $params{ccbilla1} || $params{'ccbilla1'};
    $pparams{ccbilla2} = $params{ccbilla2} || $params{'ccbilla2'};
    $pparams{cccity} = $params{cccity} || $params{'cccity'};
    $pparams{ccstate} = $params{ccstate} || $params{'ccstate'};
    $pparams{ccpostcode} = $params{ccpostcode} || $params{'ccpostcode'};
    $pparams{upgrade} = $params{upgrade} || $params{'upgrade'};
    while (($k, $v) = each %params) {
       if ($k =~ /new/i and $v ne '' and defined($v)) {
          $pparams{new} .= "$v|";
       }
    }
    chop $pparams{new};
    ModifyBillRecord(@cgi, QueryUserAccounts(@cgi, $username, 'billno'), \%pparams);
    system("./areyousure.cgi");
} elsif ($params{'cancelorder.x'}) {
    $pparams{cctype}  = '';
    $pparams{ccno} = '';
    $pparams{ccname} = '';
    $pparams{ccexp} = '';
    $pparams{billcycle} = '';
    $pparams{ccbilla1} = '';
    $pparams{ccbilla2} = '';
    $pparams{cccity} = '';
    $pparams{ccstate} = '';
    $pparams{ccpostcode} = '';
    $pparams{upgrade} = '';
    $pparams{new} = '';
    ModifyBillRecord(@cgi, QueryUserAccounts(@cgi, $username, 'billno'), \%ppar
    system("./webdesk.cgi");
  }
@| # Uncomment when handling different resellers differently.
}

sub UPS_HTML {
   my ($ups) = @_;
   my (@features, @prices, $k, $v, $d, $p, $dp, $f, @f, @f1, @f2, $m, $header, $h
   $html = '';
   $header = 0;
   @features = fo["%%DATA_INST%%/feature.dat"];
   @prices = fo["%%DATA_INST%%/price.dat"];
   while (($k, $v) = each %ups) {
      if (defined($v) and $v ne '') {
         $html .= "<TR>\n";
         $html .= "<TD COLSPAN=4 VALIGN=TOP>\n";
         @prices =~ /$v|->(.)*/;
         $dp = $&; $dp =~ /$v|->/; $dp = $';
         if ($header == 0) {
            $html .= "<HR>\n";
            $html .= "<B><FONT size=+1>Product Purchase and Upgrade Options</FONT>
            $html .= "<P>\n";
            $header = 1
         }
         ($d, $p) = split(/:/, $dp);
         $html .= "<INPUT TYPE=RADIO NAME=upgrade VALUE=\"$v\"><FONT size=+1><B>$d
h)</B></FONT>\n";
         $html .= "</TD>\n"; $html .= "</TR>\n";
         @features =~ /$v|->(.)*/;
         $f = $&; $f =~ /$v|->/; $f = $';
         @f = split(/\|/, $f);
         $m = int($#f/2);
         @f1 = @f[0..$m];
         @f2 = @f[$m+1..$#f];
         $html .= "<TR>\n";
         $html .= "<TD COLSPAN=2>\n";
         $html .= "<UL>\n";
         $html .= "<FONT size=-1>\n";
         foreach (@f1) { $html .= "<LI>$_\n" }
         $html .= "</FONT>\n";
         $html .= "</UL>\n";
         $html .= "</TD>\n";
         $html .= "<TD COLSPAN=2>\n";
         $html .= "<UL>\n";
         $html .= "<FONT size=-1>\n";
         foreach (@f2) { $html .= "<LI>$_\n" }
         if ($v =~ /(P|N)/i) {
            $html .= "<LI><a href=\"/%%MAIL_CGI_BIN%%/productinfo_webtvd.cgi\">Mo
\n";
         } elsif ($v =~ /A/i) {
```

FIG. 89



```
                $html .= '';
            } elsif ($v =~ /E/i) {
                $html .= "<LI><a href=\"/$$MAIL_CGI_BIN$$/productinfo_ee.cgi\">More Info</a>\n";
            } elsif ($v =~ /W/i) {
                $html .= "<LI><a href=\"/$$MAIL_CGI_BIN$$/productinfo_wr.cgi\">More Info</a>\n";
            }
            $html .= "</FONT>\n";
            $html .= "</UL>\n";
            $html .= "</TD>\n";
            $html .= "</TR>\n";
        }
    }
    return $html
}

sub NEW_HTML {
    my ($new) = @_;
    my ($features, $prices, $k, $v, $d, $p, $dp, $f, $f, $f1, $f2, $n, $header, $html);
    $html = '';
    $header = 0;
    $features = fo("$$DATA_INST$$/feature.dat");
    $prices = fo("$$DATA_INST$$/price.dat");
    while (($k, $v) = each %new) {
        if (defined($v) and $v ne '') {
            $html .= "<TR>\n";
            $html .= "<TD COLSPAN=4 VALIGN=TOP>\n";
            $prices =~ /$|v|->(.)*/;
            $dp = $&; $dp =~ /$|v|->/; $dp = $';
            ($d, $p) = split(/:/, $dp);
            $html .= "<INPUT TYPE=CHECKBOX NAME=\"new-$v\" VALUE=\"$v\"><FONT size=1><B>$d (\$$
p/Month)</B></FONT>\n";
            $html .= "</TD>\n"; $html .= "</TR>\n";
            $features =~ /$|v|->(.)*/;
            $f = $&; $f =~ /$|v|->/; $f = $';
            @f = split(/\|/, $f);
            $n = int($#f/3);
            @f1 = @f[0..$n];
            @f2 = @f[$n+1..$#f];
            $html .= "<TR>\n";
            $html .= "<TD COLSPAN=2>\n";
            $html .= "<UL>\n";
            $html .= "<FONT size=1>\n";
            foreach (@f1) { $html .= "<LI>$_\n" }
            $html .= "</FONT>\n";
            $html .= "</UL>\n";
            $html .= "</TD>\n";
            $html .= "<TD COLSPAN=2>\n";
            $html .= "<UL>\n";
            $html .= "<FONT size=1>\n";
            foreach (@f2) { $html .= "<LI>$_\n" }
            if ($v =~ /(P|M)/i) {
                $html .= "<LI><a href=\"/$$MAIL_CGI_BIN$$/productinfo_weftvd.cgi\">M
re Info</a>\n";
            } elsif ($v =~ /S/i) {
                $html .= '';
            } elsif ($v =~ /E/i) {
                $html .= "<LI><a href=\"/$$MAIL_CGI_BIN$$/productinfo_ee.cgi\">More Info</a>\n";
            } elsif ($v =~ /W/i) {
                $html .= "<LI><a href=\"/$$MAIL_CGI_BIN$$/productinfo_wr.cgi\">More Info</a>\n";
            }
            $html .= "</FONT>\n";
            $html .= "</UL>\n";
            $html .= "</TD>\n";
            $html .= "</TR>\n";
        }
    }
    return $html
}

sub UPG_NEW {
    my (@currentproducts) = @_;
    my (@currentproductstr, $upgrade, $service, @currentproducts, $prices, $upgrad
dp, $d, $p, $upgradeopts, $services, $upgrades, $services, %new);
    if ($currentproducts =~ /\|/) {
        @currentproducts = split(/\|/, $currentproducts)
    } else {
        push @currentproducts, ($currentproducts)
    }
    $prices = fo("$$DATA_INST$$/price.dat");
    $upgrades = fo("$$DATA_INST$$/upgrade.dat");
    %upe = ('universal', 'universal');
    my %buf;
    foreach (@currentproducts) {
        $prices =~ /$|_|->(.)*/;
        $dp = $&; $dp =~ /$|_|->/; $dp = $';
        ($d, $p) = split(/:/, $dp);
        @currentproductstr .= "$d, ";
        if ($upgrades =~ /$|_|->(.)*/) {
            $upgradeopts = $&;
            $upgradeopts =~ /$|_|->/;
            $upgradeopts = $';
            ($upgradeopts, $services) = split(/\)\(/, $upgradeopts);
            $upgradeopts =~ s/^\(//;
            $services =~ s/\)$//;
            if ($upgradeopts =~ /\|/) { @upgrades = split(/\|/, $upgradeopts) }
            else { push @upgrades, ($upgradeopts) }
            if ($services =~ /\|/) { @services = split(/\|/, $services) }
            else { push @services, ($services) }
            unless ($upe{universal} ne 'universal') {
                if ($#upgrades >= 0) {
                    foreach $upgrade (@upgrades) {
                        $upe{$upgrade} = $upgrade
                    }
                }
                $upe{universal} = '';
            } else {
                if ($#upgrades >= 0) {
                    foreach $upgrade (@upgrades) {
                        if ($upe{$upgrade} eq $upgrade) { $buf{$upgrade} = $upgrade
                    }
                }
                %upe = %buf;
                %buf = ();
                $upe{universal} = '';
            }
            foreach $service (@services) { $new{$service} = $service }
        }
        foreach (@currentproducts) { $upe{$_} = ''; $new{$_} = ''; }
    }
    $currentproductstr =~ s/,\s$//;
    return ($currentproductstr, \%upe, \%new);
}

1;
```

FIG. 90

registration.xpm

```
# -*- perl -*-
package registration;
use AddINC qw($$PERL_LIB$$);
use strict;
use CGI;
use Math::TrulyRandom;
use billing qw(GenNewBillNumber AddBillRecord ModifyBillRecord QueryBillRecord);
use reghistory qw(AddHistory);
use action qw(GenNewActionNumber AddAction);
use marketing qw(webpages);
use useraccounts qw(UserExists AddUserAccounts Encrypt PREEMPT_ON_FALSE);
use rutils  qw(fo live zf date dn dnstr emconfig @DELIM @DEBUG @TESTING);
use makeGlobalVars qw(EMAIL_ACCOUNTS);
use vars qw(@ISA @EXPORT_OK);
use Exporter();
@ISA = qw(Exporter);
@EXPORT_OK = qw(registration);

sub registration {
   my (@cgi, %params) = @_;
   my (@k, @v, @service, @transaction, @m, @sufficient, @billno, @i, @record, @encrypt, @key,
@val, @terms_file, @reglogin_file, @info_file, @pre, @match, @post);
   while (($k, $v) = each %params) {
      if ($k !~ /password/i) {
         $params{$k} =~ s/^\s+//;
         $params{$k} =~ s/\s+$//;
      }
   }
   $sufficient = 1;
   @service = @cgi->path_info;
   if ($cgi->server_name =~ /loaweb/i) {
      if ($params{control} eq '' or (not defined($params{control}))) {
         unless (defined($service) and $service ne '') {
            return undef;
         } else {
            $service = ($service =~ /^\/+/) ? $' : $service;
            $m = fo("$$LOANER_HTML_INST$$/$service_signup.html");
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => '0' } }, $m);
            print "Content-type:text/html\n\n$m"; exit;
         }
      } elsif ($params{control} =~ /continue/i) {
         $service = $params{__service__};
         $transaction = $params{__transaction__};
         while (($k, $v) = each %params) {
            if ($k =~ /\s+/ and ($v eq '' or (not defined($v)))) {
               $sufficient = 0; last;
            }
         }
         unless ($sufficient) {
            $m = fo("$$LOANER_HTML_INST$$/$service_signup.html");
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction } }, $m);
            $m = rf({\%params, $m});
            $m =~ /<\s*BODY[^>]*>/;
            $m = "$`".q{<CENTER><FONT FACE='Ariel,Helvetica'><FONT SIZE=-1><FONT COLOR=#
310039>Please fill out all the required information on this form</FONT></FONT></FONT></CENTE
R>\n".$';
            print "Content-type:text/html\n\n$m"; exit;
         } else {
            $billno = GenNewBillNumber($cgi);
            AddBillRecord($cgi, $billno, %params);
            $m = fo("$$LOANER_HTML_INST$$/$service_reg_login.html");
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction,
                                    billno      => $billno } }, $m);
            print "Content-type:text/html\n\n$m"; exit;
         }
      } elsif ($params{control} =~ /register/i) {
         $service = $params{__service__};
         $transaction = $params{__transaction__};
         $billno = $params{__billno__};
         while (($k, $v) = each %params) {
            if ($k =~ /\s+$/ and ($v eq '' or (not defined($v)))) {
               $sufficient = 0; last;
            }
         }
         unless ($sufficient) {
            $m = fo("$$LOANER_HTML_INST$$/$service_reg_login.html");
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction,
                                    billno      => $billno } }, $m);
            $m = rf({\%params, $m});
            $m =~ /<\s*BODY[^>]*>/;
            $m = "$`".q{<CENTER><FONT FACE='Ariel,Helvetica'><FONT SIZE=-1>
310039>Please fill out all the required information on this form</FONT></FONT></
R>\n".$';
            print "Content-type:text/html\n\n$m"; exit;
         } else {
            if ($params{'password1'} ne $params{'password2'}) {
               $m = fo("$$LOANER_HTML_INST$$/$service_reg_login.html");
               $m = live({ action => $cgi->script_name,
                           hidden => { service     => $service,
                                       transaction => $transaction,
                                       billno      => $billno } }, $m);
               $params{'password1'} = ''; $params{'password2'} = '';
               $m = rf({\%params, $m});
               $m =~ /<\s*BODY[^>]*>/;
               $m = "$`".q{<CENTER><FONT FACE='Ariel,Helvetica'><FONT SIZE=
R-#310039>Please retype the passwords correctly for verification</FONT></FONT></
R>\n".$';
               print "Content-type:text/html\n\n$m"; exit;
            }
            elsif (UserExists($cgi, $params{'username'}, PREEMPT_ON_FALSE)
                   == 1)
            {
               $m = fo("$$LOANER_HTML_INST$$/$service_reg_login.html");
               $m = live({ action => $cgi->script_name,
                           hidden => { service     => $service,
                                       transaction => $transaction,
                                       billno      => $billno } }, $m);
               $m = rf({}, $m);
               $m =~ /<\s*BODY[^>]*>/;
               $m = "$`".q{<CENTER><FONT FACE='Ariel,Helvetica'><FONT SIZE=
R-#310039>User Name <B>$params{'username'}</B> has already been taken, use a d
ame</FONT></FONT></FONT></CENTER>\n".$';
               print "Content-type:text/html\n\n$m"; exit;
            } else {
               $params{'password1'} = Encrypt($params{'password1'});
               $params{'pfs2dir'} = ($params{'pfs2dir'}) ? ($params{'pfs2dir'})
username'};
               AddUserAccounts($cgi, $params{'username'}, %params);
```

FIG. 91

```
                ModifyBillRecord(@billno, { status=>'0' });
$creating necessary directories
                if (sendconfig("%%LOAWEB_SENDMAIL_CONFIG_FILE%%",
                            @params{'username'}) == 0)
                {
                   @DEBUG .= "Added @params{'username'} to %%LOAWEB_SENDMAIL_CONFIG_FILE%%<b
r>\n";
                } else {
                   @DEBUG .= "Failed to add @params{'username'} to %%LOAWEB_SENDMAIL_CONFIG_
FILE%%<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}",
                        0775))
                {
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'} directory<br>\n";
                } else {
                   @DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}
 directory<br>\n";
                }

                if (-e "@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}") {
                   open(FH, ">@{[ EMAIL_ACCOUNTS{@cgi} ]} / @params{'username'}
                            / profile.cfg");
                   print FH "fullname\@@[[ QueryBillRecord( @cgi, @billno,
                                                        'firstname'} ]]",
                         "@[[ QueryBillRecord(@cgi, @billno, 'lastname'} ]]\n";
                   close(FH);
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'}/profile.cfg<br>\n";
                } else {
                   @DEBUG .= "Cannot create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/pr
ofile.cfg<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/mail", 0775)) {
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'}/mail directory<br>\n";
                } else {
                   @DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}
/mail directory<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/mail/mail_data", 07
75)) {
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'}/mail/mail_data directory<br>\n";
                } else {
                   @DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}
/mail/mail_data directory<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/mail/attachments",
0775)) {
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'}/mail/attachments<br>\n"
                } else {
                   @DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}
/mail/attachments<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/www", 0775)) {
                   @DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'usernam
e'}/www<br>\n";
                } else {
                   @DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}
/www<br>\n";
                }

                symlink "@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/www", "%%LO
AL_DIR%%/@params{'username'}";

                if (webpages{@service}) {
                   $n = fo("%%LOAWEB_HTML_INST%%/qtouchwelcome.html");
                   $n =~ s/\$username/@params{'username'}/g;
                   open(FH, ">@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/www/w
");
                   print FH $n;
                   close(FH);
                   chmod 0644, "@{[EMAIL_ACCOUNTS{@cgi}]}/@params{'username'}/www.
ml";
                }

                print "Content-type:text/html\n\n@{TESTING}"; exit;
             }
          }
       }
    } elsif ($cgi->server_name =~ /(sony|inergy)/i) {
       if (defined(@params{'terms.y'})){
          print "Content-type:text/html\n\n";
          $terms_file = "%%SONY_HTML_INST%%/terms.html";
          open (TERMS, $terms_file) || die "can't open file %%SONY_HTMLURL%%/ter m
n";
          while (<TERMS>){
             while ((@ky, @val) = each %params){
                if ($ky =~ /\t$/){chop $ky;}
                if (/\s*TYPE="hidden"\s*NAME="$ky"/){
                   $pre = $`; $match = $&; $post = $';
                   $post =~ s/VALUE="$$$$"/VALUE="$val"/;
                   $_ = "$pre"."$match"."$post";
                }
             }
             print $_;
          }
          close (TERMS);
          exit;
       }
       if (defined(@params{'accept.x'})){
          print "Content-type:text/html\n\n";
          $reglogin_file = "%%SONY_HTML_INST%%/register_login.html";
          open (REGFILE, $reglogin_file);
          while (<REGFILE>){
             while ((@ky, @val) = each %params){
                if (/<\s*INPUT\s*TYPE\s*=\s*"text"\s*NAME="$ky[\t]*"[^>]*>/){
                   $pre = $`; $match = $&; $post = $';
                   $match =~ s/>/ VALUE="$val">/;
                   $_ = "$pre"."$match"."$post";
                }
                if (/<\s*INPUT\s*TYPE\s*=\s*"password"\s*NAME="$ky[\t]*"[^>]*>/){
                   $pre = $`; $match = $&; $post = $';
                   $match =~ s/>/ VALUE="$val">/;
                   $_ = "$pre"."$match"."$post";
                }

             }
```

FIG. 92

registration.xpin

```
            print $_;
         }
         close (REGFILE);
         exit;
      }
      if ($params{control} =~ /continue/i) {
         $service = $params{__service__};
         $transaction = $params{__transaction__};
         $m = fn("%%SONY_HTML_INST%%/signup.html");
         while (($k, $v) = each %params) {
            if ($k =~ /\+@/ and ($v eq '' or (not defined($v)))) {
               $sufficient = 0;
               chop $k;
               $m =~ s/<!--@@[k]-holder@-->/<font color=greenyellow>/;
               $m =~ s/<!--@@[k]-end@-->/<\/font>/;
            }
         }


         unless ($sufficient) {
            $m =~ s/<!--@ml-holder@-->/Please complete missing information (highlighted in g
reen)./;
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction } }, $m);
            $m = rf(\%params, $m);
            print "Content-type:text/html\n\n$m"; exit;
         } else {
            my $phone = $params{'phone#'};
            $phone =~ s/[^\d]//g;
            if ($phone <= 999999999) {
               $m =~ s/<!--@ml-holder@-->/Please re-enter your phone number, including area
code and also country code if not within the US./;
               $m =~ s/<!--@phone-holder@-->/<font color=greenyellow>/;
               $m =~ s/<!--@phone-end@-->/<\/font>/;
               $m = live({ action => $cgi->script_name,
                           hidden => { service     => $service,
                                       transaction => $transaction } }, $m);
               $m = rf(\%params, $m);
               print "Content-type:text/html\n\n$m"; exit
            }
            $billno = GenNewBillNumber($cgi);
            AddBillRecord($cgi, $billno, %params);
            $m = fn("%%SONY_HTML_INST%%/register_login.html");
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction,
                                    billno      => $billno } }, $m);
            print "Content-type:text/html\n\n$m"; exit;
         }
      } elsif ($params{control} =~ /register/i) {
         $m = fn("%%SONY_HTML_INST%%/register_login.html");
         $params{__service__} =~ s/@@!/@/;
         $service = $params{__service__};
         $params{__transaction__} =~ s/PP!/P/;
         $transaction = $params{__transaction__};
         $billno = $params{__billno__};
         $params{'username#'} =~ s/^\s+//;
         $params{'username#'} =~ s/\s+$//;
         while (($k, $v) = each %params) {
            if ($k =~ /\+@/ and ($v eq '' or (not defined($v)))) {
               $sufficient = 0;
               chop $k;
               $m =~ s/<!--@@[k]-holder@-->/<font color=yellowgreen>/;
               $m =~ s/<!--@@[k]-end@-->/<\/font>/;
            }
         }
         if (!defined($params{'termsandconditions'}) or $params{'termsandconditio
n') {
            $sufficient = 0;
            $m =~ s/<!--@termsandconditions-holder@-->/<font color=greenyellow>/;
            $m =~ s/<!--@termsandconditions-end@-->/<\/font>/;
            $m =~ s/CHECKED//i;
         }
         unless ($sufficient) {
            $m =~ s/<!--@ml-holder@-->/Please complete missing information (highl:
reen)/;
            $m =~ s/<!--@password1-holder@-->/<font color=yellowgreen>/;
            $m =~ s/<!--@password1-end@-->/<\/font>/;
            $m =~ s/<!--@password2-holder@-->/<font color=yellowgreen>/;
            $m =~ s/<!--@password2-end@-->/<\/font>/;
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction,
                                    billno      => $billno } }, $m);
            $m = rf(\%params, $m);
            print "Content-type:text/html\n\n$m"; exit;
         } elsif ($params{'username#'} =~ /[\s\"@\\\/'``!#\$%\^&\(\)\+=\|\[\]\{\}:,
<
            my @forbidden_char = ();
            $m =~ s/<!--@ml-holder@-->/"@{forbidden_char}"s are not allowed in u
            $m =~ s/<!--@username-holder@-->/<font color=yellowgreen>/;
            $m =~ s/<!--@username-end@-->/<\/font>/;
            $m = live({ action => $cgi->script_name,
                        hidden => { service     => $service,
                                    transaction => $transaction,
                                    billno      => $billno } }, $m);
            $params{'username#'} = '';
            $params{'password1#'} = ''; $params{'password2#'} = '';
            $m = rf(\%params, $m);
            print "Content-type:text/html\n\n$m"; exit;
         } else {
            if ($params{'password1#'} ne $params{'password2#'}) {
               $m =~ s/<!--@ml-holder@-->/Please re-type your password carefully
d in green)/;
               $m =~ s/<!--@password1-holder@-->/<font color=yellowgreen>/;
               $m =~ s/<!--@password2-holder@-->/<font color=yellowgreen>/;
               $m =~ s/<!--@password1-end@-->/<\/font>/;
               $m =~ s/<!--@password2-end@-->/<\/font>/;
               $m = live({ action => $cgi->script_name,
                           hidden => { service     => $service,
                                       transaction => $transaction,
                                       billno      => $billno } }, $m);
               $params{'password1#'} = ''; $params{'password2#'} = '';
               $m = rf(\%params, $m);
               print "Content-type:text/html\n\n$m"; exit;
            } elsif (UserExists($cgi, $params{'username#'}, PREEMPT_ON_FALSE)) {
               $m =~ s/<!--@ml-holder@-->/"$params{'username#'}" has already been
ase choose another username!/;
               $m =~ s/<!--@username-holder@-->/<font color=yellowgreen>/;
               $m =~ s/<!--@username-end@-->/<\/font>/;
               $m = live({ action => $cgi->script_name,
                           hidden => { service     => $service,
                                       transaction => $transaction,
                                       billno      => $billno } }, $m);
               $m = rf({}, $m);
```

FIG. 93

```
                print "Content-type:text/html\n\n$m"; exit;
             } else {
                $params{'password1'} = Encrypt($params{'password1'});
                $params{'pfs2dir'} = ($params{'pfs2dir'}) ? ($params{'pfs2dir'}) : $params{'u
sername'};
                AddUserAccounts($cgi, $params{'username'}, %params);
                ModifyBillRecord($cgi, $billno, { status=>'0', 'date'=>date });
                my $action = GenNewActionNumber($cgi);
                AddAction($cgi, $action, \%params);
                AddHistory($cgi, $billno, $action);
#creating necessary directories
                if (sconfig("%%SONY_SENDMAIL_CONFIG_FILE%%", $params{'username'}) == 0) {
                   $DEBUG .= "Added $params{'username'} to %%WEB_SERVER_ROOT%%/SonyEmailUser
s.db<br>\n";
                } else {
                   $DEBUG .= "Failed to add $params{'username'} to %%WEB_SERVER_ROOT%%/SonyE
mailUsers.db<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}",
                          0775))
                {
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'} directory<br>\n";
                } else {
                   $DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}
 directory<br>\n";
                }

                if (-e "@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}") {
                   open(FH, ">@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/profile.cfg");
                   print FH "fullname\@@{[QueryBillRecord($cgi, $billno, 'firstname')]} @{[Qu
eryBillRecord($cgi, $billno, 'lastname')]}\n";
                   close(FH);
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'}/profile.cfg<br>\n";
                } else {
                   $DEBUG .= "Cannot create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/pr
ofile.cfg<br>\n";
                }

                if (mkdir("@{[ EMAIL_ACCOUNTS($cgi) ]}/$params{'username'}/mail",
                          0775))
                {
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'}/mail directory<br>\n";
                } else {
                   $DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}
/mail directory<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/mail/mail_data",
                          0775))
                {
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'}/mail/mail_data directory<br>\n";
                } else {
                   $DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}
/mail/mail_data directory<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/mail/attachments",
                          0775))
                {
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'}/mail/attachments<br>\n";
                } else {
                   $DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}
/mail/attachments<br>\n";
                }

                if (mkdir("@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www", 07
                {
                   $DEBUG .= "Successfully created @{[EMAIL_ACCOUNTS($cgi)]}/$params{'usernam
e'}/www<br>\n";
                } else {
                   $DEBUG .= "Failed to create @{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}
/www<br>\n";
                }

                open(WWW, ">@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www/wel
;
                print WWW "<html><head><title>Welcome</title></head><body><h    e
 $params{'username'} will appear here soon.</h1></body></html>";
                close(WWW);
                umask(0);
                chmod 0644, "@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www/wel
;

                symlink "@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www", "%%EM
_DIR%%/$params{'username'}";

                if (webpages($service)) {
                   $m = fc("%%SONY_HTML_INST%%/qtouchwelcome.html");
                   $m =~ s/\@username/$params{'username'}/g;
                   open(FH, ">@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www/w
");
                   print FH $m;
                   close(FH);
                   chmod 0644, "@{[EMAIL_ACCOUNTS($cgi)]}/$params{'username'}/www.
ml";
                }

                $m = fc("%%SONY_HTML_INST%%/congratulations.html");
                $m = live({ action => $cgi->script_name,
                            hidden => { username => $params{'username'} } } . $m);
                print "Content-type:text/html\n\n$m"; exit;
             }
          }
       } elsif ($params{control} =~ /webdesk/i) {
          require 'quote.pl';
          my $USERNAME = $params{'username'} || $params{'username'} || $params{'_
'};
          my $bgcolor = "#d5d1c0";
          my $frame = 1;
          my $check = truly_random_value();
          my $expire = time();
          my $profile = "%%EMAIL_ACCOUNTS%%/$USERNAME/profile.cfg";
          open(PROFILE, $profile);
          my $REALNAME = <PROFILE>;
          close(PROFILE);
          chop $REALNAME;
          $REALNAME =~ s/fullname\@//;
          my $checkfile = "%%EMAIL_ACCOUNTS%%/$USERNAME/mail/.check";
          open(CHECKFILE, ">$checkfile") || die "Unable to write check file $check:
          print CHECKFILE "$expire:$check:$REALNAME:$USERNAME\n";
```

FIG. 9d

registration.xpm

```
        close(CHECKFILE);
        $| = 1;
        print("Set-Cookie: username=$USERNAME; path=/\n");
        print("Set-Cookie: check=$check; path=/\n");
        $ENV{'HTTP_COOKIE'} = "username=$USERNAME&check=$check";
        $cgi->delete('password');
        $cgi->delete('submit');
        $ENV{'QUERY_STRING'} = $cgi->query_string;
        if ($frame && !$cgi->param('frame')) {
           $ENV{'QUERY_STRING'} .= "&frame=1";
        }
        $ENV{'REQUEST_METHOD'} = "GET";
        $ENV{'SCRIPT_NAME'} = "webdesk.cgi";
        my $quota_used = check_dir("$$EMAIL_ACCOUNTS$$/$USERNAME");
        if ($quota_used > 10000000) {
           print "\n<META HTTP-EQUIV=\"Refresh\" CONTENT=\"3;URL=/$$MAIL_CGI_BIN$$/webdesk.
cgi?username=$USERNAME&frame=$frame\" >";
           print "<BODY bgcolor=\"$bgcolor\">";
           print "<center><BR><BR><font size=3 face=arial>You have crossed your quote size<
/t><br>";
           print "</center></body></html>";
           exit(0);
        } else {
           system("./webdesk.cgi");
           exit(0);
        }
        print("Content-type: text/plain\n\n");
        print("Exec failed!\n");
     }
```

FIG. 95

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/00 | A3

(11) International Publication Number: **WO 99/09658**

(43) International Publication Date: 25 February 1999 (25.02.99)

(21) International Application Number: PCT/US98/16894

(22) International Filing Date: 14 August 1998 (14.08.98)

(30) Priority Data:

60/055,782	15 August 1997 (15.08.97)	US
60/057,256	29 August 1997 (29.08.97)	US
60/060,612	1 October 1997 (01.10.97)	US
08/970,894	13 November 1997 (13.11.97)	US
60/065,416	13 November 1997 (13.11.97)	US
08/971,002	14 November 1997 (14.11.97)	US

(71) Applicant: INERGY ONLINE, INC. [US/US]; 20 Mall Road, Burlington, MA 01803 (US).

(72) Inventor: BELANGER, Charles, E.; 345 Boston Road, Billerica, MA 01821 (US).

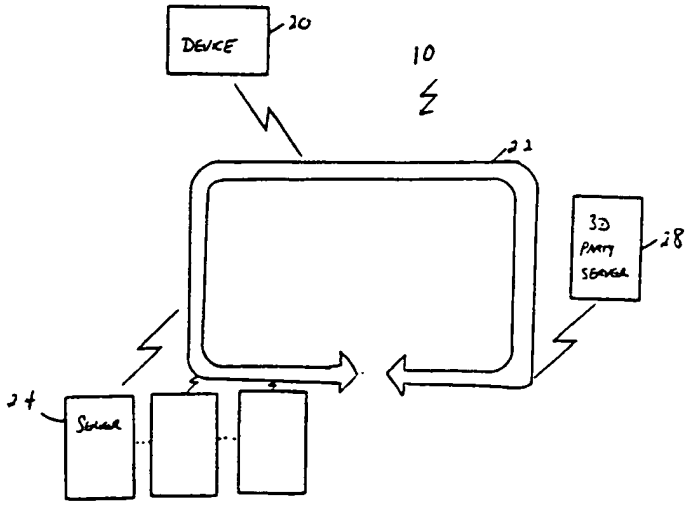
(74) Agents: CELLA, Charles, H. et al.; Foley, Hoag & Eliot LLP, One Post Office Square, Boston, MA 02109 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(88) Date of publication of the international search report:
13 April 2000 (13.04.00)

(54) Title: SERVER-SIDED INTERNET-BASED PLATFORM INDEPENDENT OPERATING SYSTEM AND APPLICATION SUITE



(57) Abstract

The systems and methods described herein provide different types of web authoring, web site management, and communication software technology, including but not limited to full multimedia authoring, online libraries, sounds, forms, e-mail, facsimile, voice-mail, pager, telephone, financial management, true document printing (as opposed to screen printing), text-to-voice and voice-to-text conversion, file management, spreadsheets, all accessed and run via the Internet (22). The system resides entirely on an Internet web server site (24) and interacts with users (20) via conventional progamming languages written for a universal protocol. As a result, there is no need for client-side messaging software. All software is provided on the server side (24). The only software the user (20) needs is any form of communication module and an electronic communication connection. Because the system (10) is platform and operating system independent, a user (20) may author, create, maintain, send and receive messages from any platform, using any conventional operating system. A user (20) may customize their desktop configuration and may run a variety of different applications. Moreover, a user (20) may switch between applications, and transfer text, graphics, or sound files between applications.